

Precio: 6 € (España) (IVA incluido) • AÑO XIV. 2.ª ÉPOCA • Nº 161 • UNA PUBLICACIÓN DE: REVISTAS PROFESIONALES S.L.

JAVAHISPANO
Actualidad Java

REDES
RSS con Java (II)
Desarrollo de una aplicación Java distribuida

DISEÑO
Programando en Java la Web Semántica con Jena (IV)

MIDDLEWARE
Programación con múltiples hilos
ASP.NET (MVC)

VÍDEO-TUTORIAL
Tiro a diana

DE REGALO
Solo Programadores número 159 en formato pdf

Y además... Solo Programadores 159 en formato pdf
Vídeos e información sobre JavaCup

SCRUM

Scrum es un framework de desarrollo, más que un proceso. En Scrum las indicaciones se limitan a “qué” se debe hacer, no entrando a dictar el “cómo”.

8 423793 003751 00161

Noticias, Opinión, Actualidad y CD-ROM



4D
4D v11 SQL
Estallido de pura potencia
4D v11 SQL
4D Developer Professional
Develop powerful database and Web applications
www.4D.com
4D v11 SQL supone la nueva era de la programación. 4D concentra en una única solución toda la innovación, sencillez y ergonomía en la creación de potentes bases de datos y aplicaciones web para todos los ámbitos profesionales.
Esta revolucionaria plataforma 4D v11 SQL ofrece nuevos niveles de escalabilidad, multiplica el rendimiento y simplifica el desarrollo de aplicaciones profesionales y de Internet. Integra a la perfección todos los elementos esenciales para el programador, permitiendo la interconexión con otras tecnologías y respetando los estándares de la industria.
El universo 4D se expande sin límites en la generación de poderosas aplicaciones finales para Mac y Windows, poniendo a tu disposición infinitas opciones para el despliegue de software. Experimenta su facilidad de mantenimiento y cómo se adapta a tus necesidades más exigentes.
4D v11 SQL constituye una apuesta brillante por las tecnologías del futuro. Con 4D v11 SQL descubrirás el impacto de la potencia pura en el rendimiento de tus aplicaciones.
4D Hispano . Tlf +34 93 665 76 72 . www.4dhispano.com . info@4dhispano.com



Número 161

Edita: REVISTAS PROFESIONALES S.L.

solop@revistasprofesionales.com

C/Valentin Beato 42, 3ª 28037 - Madrid

www.revistasprofesionales.com

Editor

Agustín Buelta

Coordinación Técnica-Redacción

Ricardo Álvarez

Colaboradores

Abraham Otero, Juan Martos, Jorge Rubira, Guillem Alsina, Adolfo Aladro, Diego López, Nicolás Velasquez, Erich Bühler, Gastín Hillar, Javier Holguera, Nacho Verdú

Maquetación

Alfonso Sabán / Raúl Clavijo

Departamento de Publicidad

Felipe Ribagorda

Tel.: 91 304 87 64

Delegación en Barcelona

C/ Rocafort, 241/243, 5º 1ª

Mariano Sánchez

Tel.: 93 322 12 38

Dpto. Suscripciones

Tel: 91 304 87 64

Fax: 91 327 13 03

Impresión

L.M.S. Solución Gráfica

ideasimpresion@telefonica.net

Distribución

LOGINTEGRAL

Saturnino Calleja, 7
28002 Madrid
Tfno. 915 864 933

DISTRIBUCION EN MEXICO

DIMSA - C/ Mariano Escobedo, 218
Col. Anáhuac. 11320 México, D.F.

DISTRIBUCION EN ARGENTINA

Capital Federal: Distrimachisa
Interior: York Agencysa - Tlf: (5411) 433 150 51

Depósito legal: M-26827-1994

PRINTED IN SPAIN

P.V.P. 6 euros

EDITORIAL

Tras el éxito obtenido el año pasado, esperamos que la edición de este año de JavaCup siga la misma línea mejorando los resultados anteriores. Los premios han aumentado y esperamos que también lo haga el número de participantes.
Coincidiendo con el concurso, hemos potenciado en los últimos números los artículos en torno a Java (como habrán podido comprobar los lectores), situación temporal que dará paso a contenidos de distinta índole en los próximos números.
En atención a sugerencias enviadas por los lectores, trataremos temas relativos a PHP, Delphi, C, C++, ActionScript, Directx y otros.
Como siempre, cualquier sugerencia al respecto será bien recibida y tenida en cuenta a la hora de confeccionar la lista de posibles contenidos.

SUMARIO



Adobe presenta el Open Screen Project



Constituye una plataforma para la presentación de contenidos multimedia interactivos en multitud de dispositivos cómo teléfonos móviles, televisores u ordenadores.

Adobe ha impulsado[1] una gran alianza junto a compañías de la talla de LG, Intel, Cisco, Sony Ericsson, Nokia o Verizon Wireless y el soporte de productores de contenidos cómo MTV, BBC o NBC Universal para proporcionar a programadores y usuarios un entorno multiplataforma y rico en el que distribuir y reproducir contenidos multimedia (audio, vídeo y aplicaciones) basado en sus productos y abierto.

La parte que ha sido considerada por la comunidad internauta cómo la más interesante es que para llevar a cabo su proyecto, Adobe va a abrir ciertas partes de la especificación de su tecnología Flash, cómo los formatos de fichero y el protocolo de comunicaciones empleado, además de eliminar las licencias, lo que dejará a los entornos de ejecución RIA de la compañía (Flash Player y AIR) cómo productos gratuitos para dispositivos móviles.

La idea es proporcionar un runtime environment, un entorno de ejecución de aplicaciones que permita a los internautas trabajar con programas completos a través de un navegador web, algo parecido a lo que Apple ofrece en el iPhone y el iPod Touch con sus aplicaciones web.

Los motivos de Adobe para lanzar ahora esta iniciativa podrían encontrarse en el éxito que está consiguiendo el Silverlight de Microsoft, un rival directo de la tecnología Flash disponible para Windows y Mac OS X, al que Adobe por el momento no debe temer, pero sí asegurarse el futuro a largo plazo para Flash.

[1] http://www.adobe.com/aboutadobe/pressroom/pressreleases/200804/050108AdobeOSP.html

Spam, 30 años

Hace tres décadas se envió el primer mensaje publicitario no solicitado por correo electrónico. Hoy el llamado "correo basura" llega a cotas del 90% en el tráfico de Internet.

Mucho ha llovido desde los tiempos en que ARPAnet[1] dejó de serlo para cambiar de nombre y pasar a ser conocida cómo Internet, una "carretera" de la información más que la actual autopista que es. Desde entonces la también llamada Red de redes ha perdido la inocencia que caracterizó sus primeros balbuceos y se ha ido enfrentando paulatinamente a crackers, virus, spam y otras tretas modernas cómo el phishing. El correo electrónico no solicitado con fines publicitarios de los más serios o disparatados productos, también conocido cómo Spam y que constituye uno de los principales problemas de la Internet moderna, cumple ahora tres décadas de existencia.

El considerado cómo primer mensaje que entra en la categoría de Spam fue enviado el 3 de Mayo de 1978 por Gary Thuerk[2], por aquel entonces director de marketing de Digital Equipment Corp. (DEC). Internet aún se conocía cómo ARPAnet y arrastraba su origen militar. Thuerk quiso invitar a todos los usuarios de ARPAnet de la costa oeste de los Estados Unidos (en aquella época unos 600, principalmente científicos) a un evento de presentación de las nuevas computadoras de su compañía, así que decidió no mandar un mensaje individualizado sino tener una plantilla genérica a enviar.

Entre él y sus compañeros de trabajo tecleaaron todas las direcciones que constaban en la guía de ARPAnet (en aquella época impresa en papel cómo la guía telefónica) y mandaron los mensajes. Thuerk fue amonestado por las molestias ocasionadas, pero consiguió ventas -lo principal en cualquier empresa-, por lo que el método se dio por bueno. Sin saberlo, había abierto la Caja de Pandora que dos décadas después empezaría a afectar seriamente a la Red y que actualmente es unánimemente considerada cómo una de las principales lacras de Internet.

El nombre de Spam procede de un famoso 'sketch' televisivo del grupo cómico británico Monty Python, emitido por primera vez en 1970 y en el cual dos viajeros intentan encargar el desayuno en un restaurante. La camarera es la encargada de cantarles la carta, y en cada plato consta una cantidad significativa de 'Spam', una carne precocinada y enlatada muy popular en Inglaterra. El grito de "¡spam!, ¡spam!, ¡spam!" lanzado por la camarera (en realidad el actor Terry Jones) fue así identificado con algo anodino y empalagoso, cómo el Spam electrónico. Pero el término no fue utilizado en el mundillo tecnológico hasta mediada la década de los ochenta del siglo pasado, en la que se denominó Spam a la práctica de usuarios de BBS's y MUD's (dos formas de comunicarse mediante computadoras que decayeron con el advenimiento de Internet) de poner trabas a otros usuarios de los mismos servicios a los que querían expulsar mediante la repetición constante de un determinado texto.

A mediados de los noventa el correo electrónico comercial no solicitado irrumpió con fuerza en los buzones de los internautas, cuyo número aumentaba entonces muy rápidamente en todo el hemisferio occidental. El término Spam no tardó en ser asignado a esta práctica concreta, y hoy en día no se identifica con algo que no sean las comunicaciones electrónicas no solicitadas con interés comercial, mayoritariamente por correo electrónico si bien se están dando cada vez con más frecuencia casos de Spam a través de sistemas de mensajería instantánea o de los comentarios en los blogs. El futuro próximo nos depara nuevas formas de Spam cómo aquellas que se darán en los teléfonos móviles.



Más información:

Artículo en la web de los Monty Phyton sobre los treinta años de Spam
http://pythonline.com/world_celebrates_30_years_of_spam_0

Sketch original del Spam en YouTube
http://www.youtube.com/watch?v=anwy2MPT5RE

[1] http://en.wikipedia.org/wiki/ARPAnet

[2] http://en.wikipedia.org/wiki/Gary_Thuerk

Psystar vende los primeros Open Computer y el mundo tecnológico los analiza con lupa

Las primeras imágenes y vídeos han sido colgadas en la Red mientras continúan las dudas sobre la validez legal de la aventura iniciada por Psystar.

La compañía que hace unas semanas acaparó las portadas de los principales medios de comunicación especializados con su anuncio en el que ofrecía clones no oficiales de los Apple Mac capaces de ejecutar el





Mac OS X para plataforma Intel en sus últimas versiones, ha empezado a proporcionar las primeras unidades a los compradores y a conocidos sitios web que empiezan a testearlas y a poner sus resultados en Internet. Uno de estos sitios es Gizmodo, un conocido sitio sobre gádgets editado en los Estados Unidos, que ha colgado[1] en su sitio un vídeo enviado por un lector, que ha sido uno de los primeros compradores de un Open Computer que ha recibido la computadora.

En el vídeo se muestran en primer lugar las conexiones del cableado en la parte trasera de la máquina, con especial énfasis en el cable de vídeo a modo de prueba que la imagen proviene realmente del Open Computer y no de un Macintosh conectado funcionando "por detrás". Seguidamente, lo que vemos es el proceso de inicio de la computadora hasta cargar la versión 10.5.2 de Mac OS X.

Un simple comentario adorna la imagen en movimiento: imposible realizar la actualización del sistema, un problema del que ya se informaba en el sitio de Psystar y que constituye uno de los handicaps de hackear un Mac OS X para que funcione en una máquina x86 genérica. No dejamos a Gizmodo, porque éste mismo sitio web muestra[2] las interioridades del Open Computer, un "strep tease" en el que puede verse la caja abierta junto a una preocupante aseveración: el Mac OS X no puede reinstalarse si le pasa algo al disco duro. Esto es debido a que con la máquina se incluye una licencia legítima del sistema operativo, pero claro, sin el hackeo necesario para hacer que funcione en la máquina no Apple. El limbo legal en el que se encuentra la operación emprendida por Psystar juega una mala pasada a los posibles compradores que atraídos por la posibilidad de ejecutar el OS X de Apple por una fracción de lo que cuesta un Macintosh original, puedan encontrarse con algún problema de tipo lógico o físico en el disco duro. Y en ese caso ¿qué? Pues probablemente sólo quede la opción de buscar en Internet, y acudir a los foros del OSX86Project[3] para que alguna alma caritativa pueda solucionar el problema. La pregunta es si los errores debidos al software serán o no frecuentes.

La información se confirma en otra de las notas sobre el "asunto Psystar" publicada[4] por Gizmodo, en la cual se explica la respuesta que la compañía fabricante del Open Computer dio a uno de los lectores de Gizmodo que preguntó sobre el tema de la reinstalación, y es que ellos no responden de los intentos llevados a cabo por los mismos usuarios debido a la dificultad del proceso.

En ZDNet han intentado ir un paso más allá identificando[5] el hack utilizado para facilitar la instalación de Leopard en las máquinas. Sería el conocido cómo Kalyway, y necesitaría de una imagen de DVD manipulada y, por lo tanto, ilegal para su instalación.

Las primeras informaciones del Open Computer no han hecho disminuir la curiosidad generada alrededor de estas máquinas y de la empresa que las ha creado y vende, más bien al contrario; se han generado nuevas preguntas y a buen seguro que el interés irá "in crescendo" durante las próximas semanas.

[1]http://gizmodo.com/384526/exclusive-video-psystar-in-the-wild
[2] http://gizmodo.com/384854/exclusive-photos-psystars-case-shipping-contents
[3] http://www.osx86project.org/
[4] http://gizmodo.com/384302/psystar-wont-let-you-reinstall-leopard-by-yourself
[5] http://blogs.zdnet.com/Apple/?p=1632

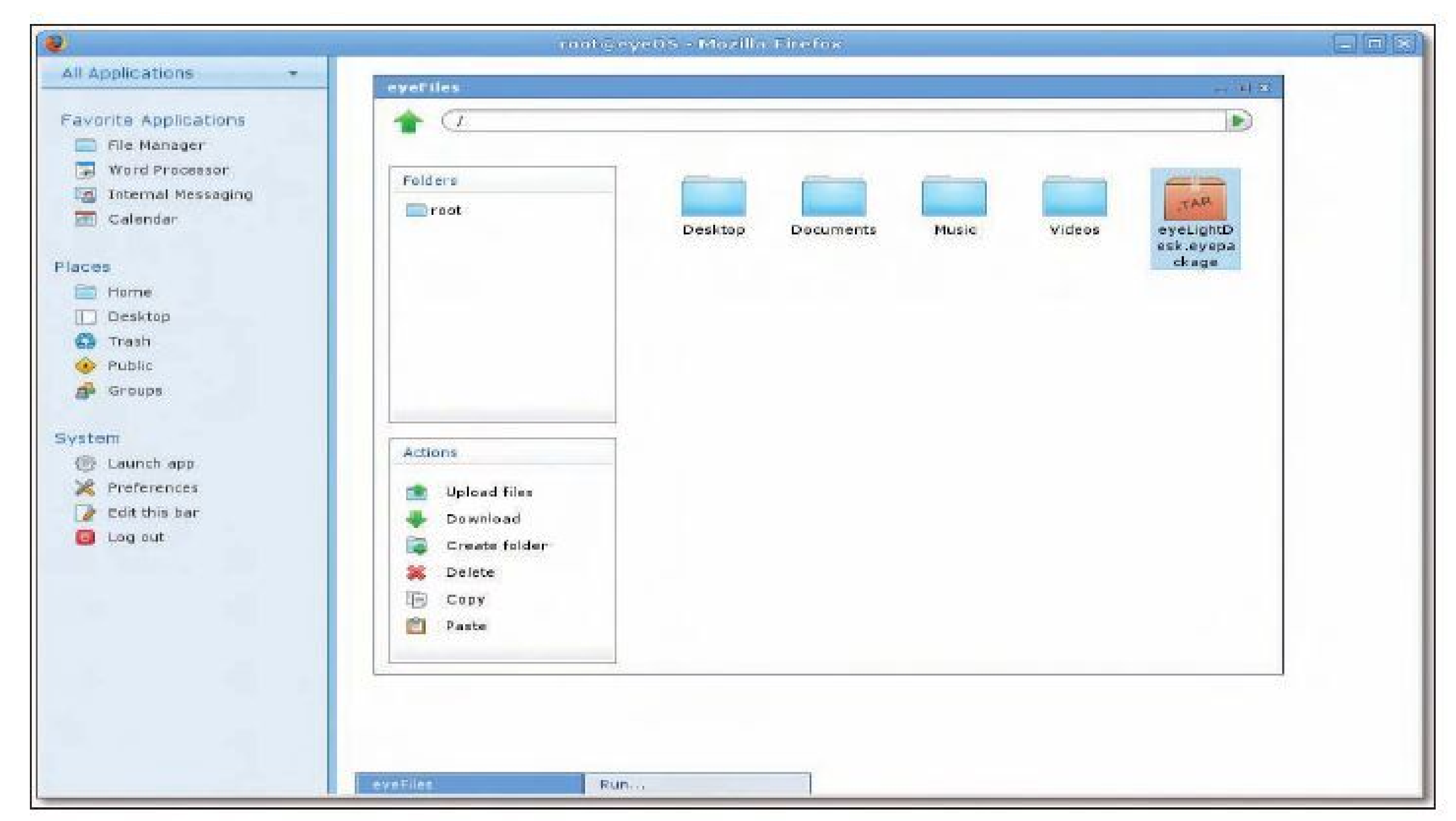
eyeOS 1.6 permite sincronizar los archivos del sistema con la computadora local

Una aplicación externa y dependiente de plataforma permite mantener sincronizados los archivos de nuestra máquina local con el conocido sistema operativo online sin necesidad de intervención explícita del usuario.

Las aplicaciones web están de moda, esto no puede negarlo nadie; Google triunfa con sus servicios Gmail, Docs&Spreadsheets o Reader entre otros para desesperación de Microsoft, que intenta seguirle el rastro con sus servicios Live. eyeOS[1] va un paso más allá, transportando todo el sistema operativo y sus funcionalidades a la web.

Con simplemente un navegador web estándar (Firefox, Internet Explorer, Safari,...) podemos acceder a todo un sistema operativo completo con su correspondiente sistema de ficheros, escritorio y aplicaciones, trabajar desde cualquier computadora y cerrar la sesión de trabajo pudiendo reabrirla en el mismo punto en el que estábamos y con el mismo entorno desde cualquier otra computadora aunque estemos en la otra punta del mundo. La disponibilidad de nuestros ficheros allá donde nos encontremos es, sin lugar a dudas, un atractivo importante que explica el éxito de éste proyecto único en su especie y que ha sido llevado a cabo por unos jóvenes catalanes sin ningún apoyo y que lo han llegado a convertir en uno de los proyectos de referencia de la Web 2.0 .

En su versión 1.6[2], eyeOS presenta varias novedades interesantes cómo una nueva aplicación para la lectura de feeds RSS (eyeFeeds), una nueva versión de Text Editor (el editor de textos estilo Notepad del Windows) con resaltado de sintaxis para los







lenguajes de programación más conocidos, un IDE (entorno de desarrollo integrado por sus siglas en inglés) llamado eyeCode ejecutable desde la cuenta de root, o nuevos juegos.

Pero sin lugar a dudas, la novedad más destacable proviene de un software oficial aunque externo al sistema operativo online y dependiente de plataforma: la posibilidad de sincronizar los archivos que tenemos en nuestra cuenta de eyeOS con la máquina local en la que estemos trabajando.

Esta aplicación, bautizada cómo eyeSync, se instala en local y, por lo tanto, depende del sistema operativo sobre el que corramos el navegador web con el cual accedemos a eyeOS. Por el momento se encuentra disponible para Mac OS X (plataformas PowerPC e Intel) y Linux, y pronto estará a disposición de los usuarios de Windows XP/Vista.

El funcionamiento de éste software una vez instalado es simple, y no va a requerir intervención por parte del usuario para sincronizar los archivos, sino que detectará los cambios que se hayan realizado y llevará a cabo las acciones de sincronización oportunas. eyeSync es una aplicación gratuita y puede descargarse desde el sitio web del proyecto al igual que eyeOS.

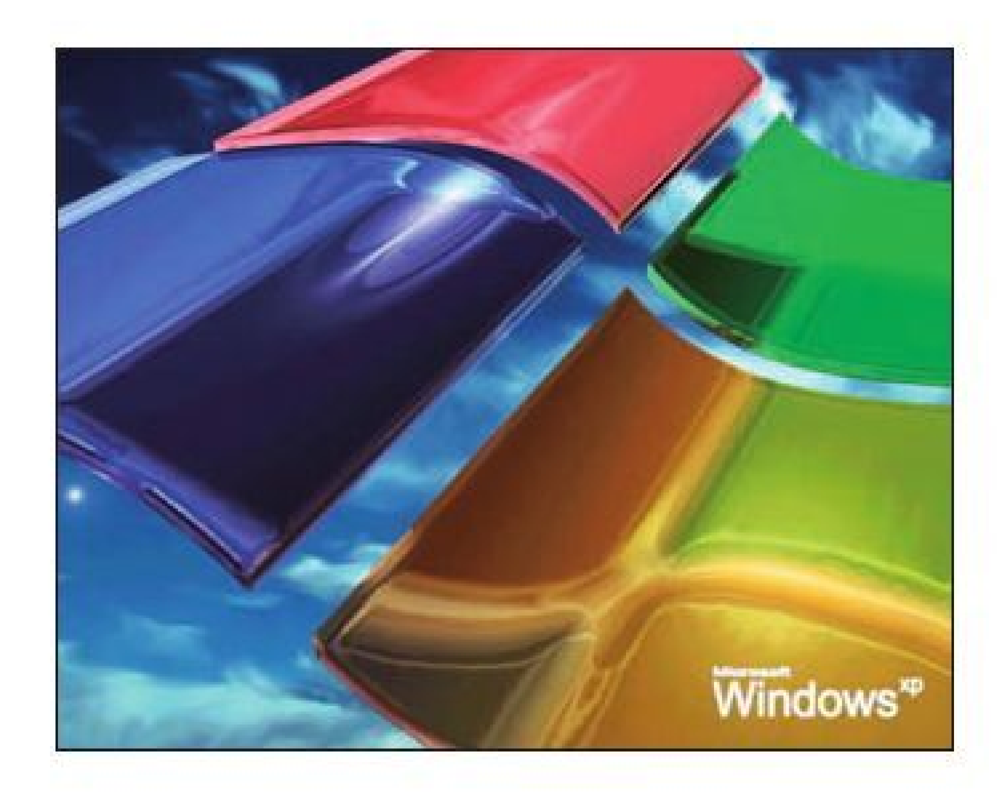
Éste sistema operativo puede ser descargado e instalado en un servidor para dar servicio a una red local o bien utilizado gratuitamente a través de Internet con sólo dar de alta una cuenta[3] en el servidor de demostración.

[1] http://eyeos.org/es/?section=home
[2] http://blog.eyeos.org/2008/04/25/eyeos-16-gala-sync-released/
[3] http://www.eyeos.info/`

Los vendedores de computadoras buscan la forma de alargar la vida a Windows XP

Con el 30 de Junio cómo horizonte final para dejar de vender el ya veterano sistema operativo de Microsoft, compradores y vendedores de computadoras leen la letra pequeña de la licencia de usuario de Windows Vista para encontrar algún resquicio legal con el cual poder hacerse con una copia de XP pasada la fecha límite. El downgrade puede ser una solución.

Pese a que Microsoft alargó el periodo de disponibilidad comercial de Windows XP hasta el próximo 30 de Junio, éste parece que no va a ser suficiente para muchos. Es, si más no, curioso, teniendo en cuenta que en su introducción en el mercado, XP padeció los mismos problemas de aceptación y las críticas que ahora está sufriendo Vista. Pero después de dos service packs (pronto serán tres), el veterano sistema operativo de la compañía creada por Bill Gates se ha ganado fama de estabilidad y efectividad.



Así es lógico que se persiga su continuidad. Y para hacerlo, para mantener la posibilidad de obtener una computadora nueva a partir de la fecha límite establecida por Microsoft, compradores y vendedores buscan los resquicios que legalmente se lo permitan. No es que a partir del 30 de Junio se prohíba la venta de Windows XP en todo el mundo, pero es que la compañía de Redmond va a dejar de proporcionar licencias nuevas para éste sistema.

Según publica el sitio web OS News[1], Hewlett-Packard y Dell son dos de las empresas interesadas en mantener activo el negocio del Windows XP, y para ello se amparan en una de las cláusulas de uso de Windows Vista, que permite continuar utilizando una versión antigua de Windows hasta que el usuario se vea capacitado para trabajar con la versión más reciente, siempre y cuando sea un usuario final.

Es lo que se conoce cómo downgrade, e implica la adquisición de una licencia de Windows Vista aunque esta no vaya a ser utilizada. La petición y todo el proceso de reinstalación debe hacerlo el usuario, pero - y aquí es donde se encuentra el truco- las dos empresas que he mencionado ofrecen cómo servicio al cliente la realización de estos pasos, algo que pueden hacer de forma legítima y que además el texto de la licencia les deriva esta tarea de Microsoft a ellos, de forma que se estará adquiriendo una licencia de Windows Vista pero en realidad se instalará y utilizará Windows XP. Incluso se ha creado una nueva denominación para éste tipo de máquinas: pre-upgrade.

Solamente las versiones Business y Ultimate disponen de esta cláusula en su licencia de uso, y el "downgrade" puede hacerse a XP Professional, la versión Home no admite esta posibilidad. Además, hay otro inconveniente, que no es más que la laguna legal en la que queda el tema de si la licencia del sistema XP debe proporcionarla el usuario (y por lo tanto haberla adquirido previamente) o bien la puede poner el vendedor siendo proporcionada por el fabricante (Microsoft).

Otra opción a Windows Vista es la instalación de la versión más básica de Server 2003, la versión profesional de Windows para máquinas servidoras, y que con el aumento de prestaciones en el hardware actual, podría funcionar bien cómo sistema operativo de sobremesa.

[1] http://www.osnews.com/story/19685/Downgrade_Rights_As_a_Backdoor_to_Continue_to_Sell_XP_

A partir del 30 de Junio ya no se podrán consultar las cuentas de Hotmail a través de Outlook Express

La compañía de Redmond ha anunciado un cambio del protocolo de comunicación empleado por Hotmail para enviar los mensajes a un programa cliente de correo electrónico, que dejará a Outlook Express sin poder comunicarse con el servicio de webmail gratuito. La compañía de Redmond no pondrá remedio a esto para forzar la migración de los usuarios a Windows Live Mail.

De cuando en cuando, las empresas de





informática realizan cambios radicales en sus productos que cortan de raíz con lo que habían ofrecido hasta la fecha y fuerzan a los usuarios a una migración masiva. Apple es el paradigma de estas prácticas, migrando de plataforma dos veces (de Motorola 60xxx a PowerPC y de esta última a Intel x86) y de sistema (del Mac OS Classic al Mac OS X basado en Unix).
Microsoft ahora realiza una de estas migraciones "traumáticas" para el usuario (aunque a pequeña escala) con la anulación del uso del protocolo DAV para transmitir los mensajes de correo almacenados en el servidor al cliente local. DAV se había quedado anticuado para las necesidades crecientes de los usuarios de Hotmail, y más con la rápida ampliación del espacio de almacenamiento proporcionado por Microsoft, lo que comporta a su vez un uso más extensivo por parte de los usuarios.
DAV dejará de funcionar el próximo 30 de Junio y será substituido por DeltaSync, un protocolo desarrollado por la propia Microsoft e integrado en sus nuevos productos de mensajería cómo Windows Live Mail, el cliente gratuito que substituye a Outlook Express en los entornos domésticos. Según la compañía de Redmond, éste protocolo proporciona una mejor gestión de carpetas de correo de grandes dimensiones.
Es precisamente la migración a éste último lo que recomienda Microsoft para todos los usuarios de Outlook Express que consulten desde éste cliente sus cuentas de Hotmail. Windows Live Mail es gratuito y puede instalarse mediante el asistente de instalación de Windows Live, con el que además pueden instalarse otros productos de la familia Live cómo Messenger.

Más información:

Nota publicada por Microsoft en el blog de soporte técnico de Windows Live Mail
http://emailsupport.spaces.live.com/Blog/cns!5D6F5A79A79B6708!5359.entry

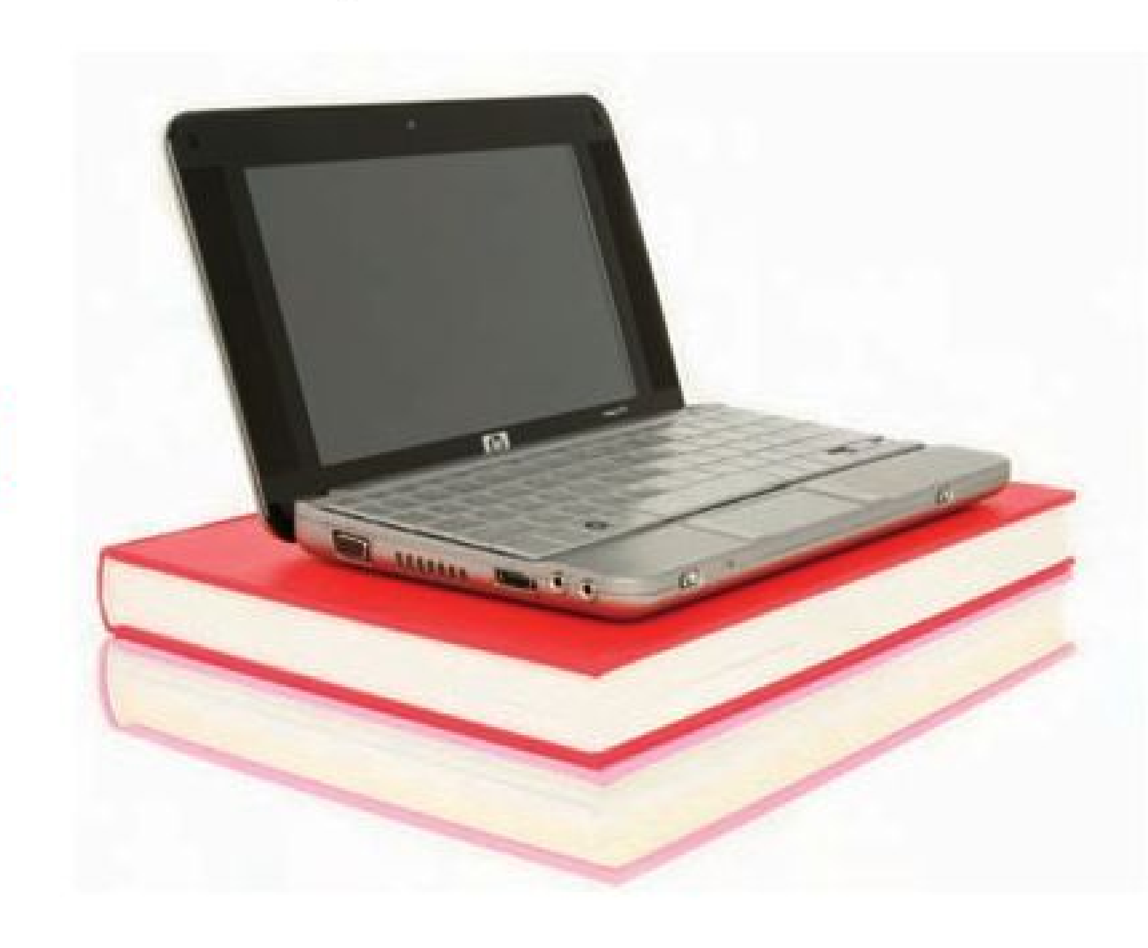
vLite, una herramienta para crear DVD's de instalación desatendida de Windows Vista a medida



Esta utilidad permite crear nuestro medio de instalación de Windows Vista completamente a medida, eliminando componentes o añadiendo drivers.
Desde que las distintas distribuciones GNU/Linux empezaron a incluir entre sus paquetes diversas utilidades para personalizar la instalación e incluso construir medios de instalación a medida cómo CD's o DVD's, diversos hackers han querido llevar a cabo lo mismo pero para el sistema Windows de Microsoft.
vLite[1] es una de estas últimas herramientas, que trabaja sobre Windows Vista para permitirnos construir un medio de instalación totalmente personalizado. Entre las facilidades que nos permite tenemos la eliminación de componentes que se instalan por defecto junto con el sistema operativo, la inclusión de packs de lengua (cómo el catalán o el gallego, idiomas solamente disponibles mediante un parche que debe ser aplicado después de la instalación en otro idioma, en estos dos casos el castellano), integración de drivers, y opciones para instalación desatendida.
La adición de drivers nos permite instalar Windows Vista de forma que una vez finalizado el proceso de instalación tengamos nuestra computadora completamente funcional, habiendo reconocido el sistema nuestros periféricos cómo la tarjeta de red inalámbrica o la tarjeta de vídeo. También podemos dividir el medio original de instalación del sistema, el DVD, en varios CD's, útil si queremos experimentar con la instalación en máquinas un poco antiguas o con ciertas limitaciones.
La descarga de esta utilidad es completamente gratuita, con un "peso" que no llega a los dos megabytes. Actualmente va por la versión 1.1.6.
[1] http://www.vlite.net/

HP lanza su ultraportátil

En cuatro configuraciones diferentes con mayor o menor potencia, monta SUSE Linux o Windows Vista sobre un procesador VIA C7 Mobile.
Hewlett-Packard ha seguido los pasos de Asus y su exitoso Eee PC con un miniportátil de muy reducidas dimensiones y de elegante aspecto profesional, marca característica de la casa.
El HP 2133 Mini-Note PC[1] tiene un peso que no llega ni al quilogramo, monta un procesador VIA con una frecuencia de reloj de entre 1 y 1,6 GHz. según versión[2], de 512 MB. a 2 GB. de memoria, pantalla de 8,9 pulgadas con una resolución de 1280x800 px., tarjeta de vídeo VIA Chrome 9 UMA, y tarjeta inalámbrica Broadcom 4311 802.11 b/g. La unidad de almacenamiento principal va desde los 4 GB. de la memoria flash de la configuración más básica hasta los 120 GB. de disco duro de la más completa. También se incluye una webcam integrada, un accesorio cada día más indispensable en esta categoría de máquinas.
Las posibilidades de personalización de la configuración de nuestra máquina son muy amplias. Sin llegar a poder montar la computadora completamente a medida, tenemos a nuestro alcance un gran abanico de variaciones que incluyen el sistema operativo, con SUSE Linux Enterprise Desktop 10 en sus dos configuraciones más básicas, Windows Vista Home Basic







en la siguiente y Windows Vista Business en la más potente.
El precio también sube en función de la potencia de los componentes y del sistema operativo. Así, la versión más económica es la que cuenta con el procesador de 1 GHz., 512 MB. de RAM, 4 GB. de memoria flash y utiliza Linux, saliendo a 499 dólares. En cambio, la versión más cara es la que monta Windows Vista Business en un disco duro de 120 GB., 2 GB. de RAM, procesador a 1,6 GHz., Bluetooth, y batería de seis celdas en lugar de la de tres del resto de los modelos, saliendo a 749 dólares. Tal vez esto precisamente, el precio, sea lo más criticable de esta máquina, elevado si lo comparamos con el del Eee PC a modo de máquina de referencia: 500 dólares el Mini-Note de HP contra unos 300 de la máquina de Asus. Tal vez sea algo excesivo que eche para atrás a más de un posible comprador.
El aspecto del Mini-Note es impecable, haciendo gala del mejor diseño de la compañía. Obviamente, su orientación comercial no es la misma que la del Eee PC (algo que ya deja claro con el precio, cómo antes he comentado), dirigiéndose a un mercado más empresarial, a ejecutivos que deban viajar mucho y aligerar peso sea una prioridad, y a estudiantes de ciclos secundarios
Para HP esta no es la primera experiencia en máquinas ultraportables, pues la compañía norteamericana cuenta en su haber con el bagaje proporcionado por la ahora extinta serie Jornada, un híbrido entre PDA y portátil llamado handheld que ofrecía en un formato de portátil con teclado QWERTY pero mucho más pequeño, una máquina basada en Windows CE con capacidad de encendido instantáneo (instant-on).

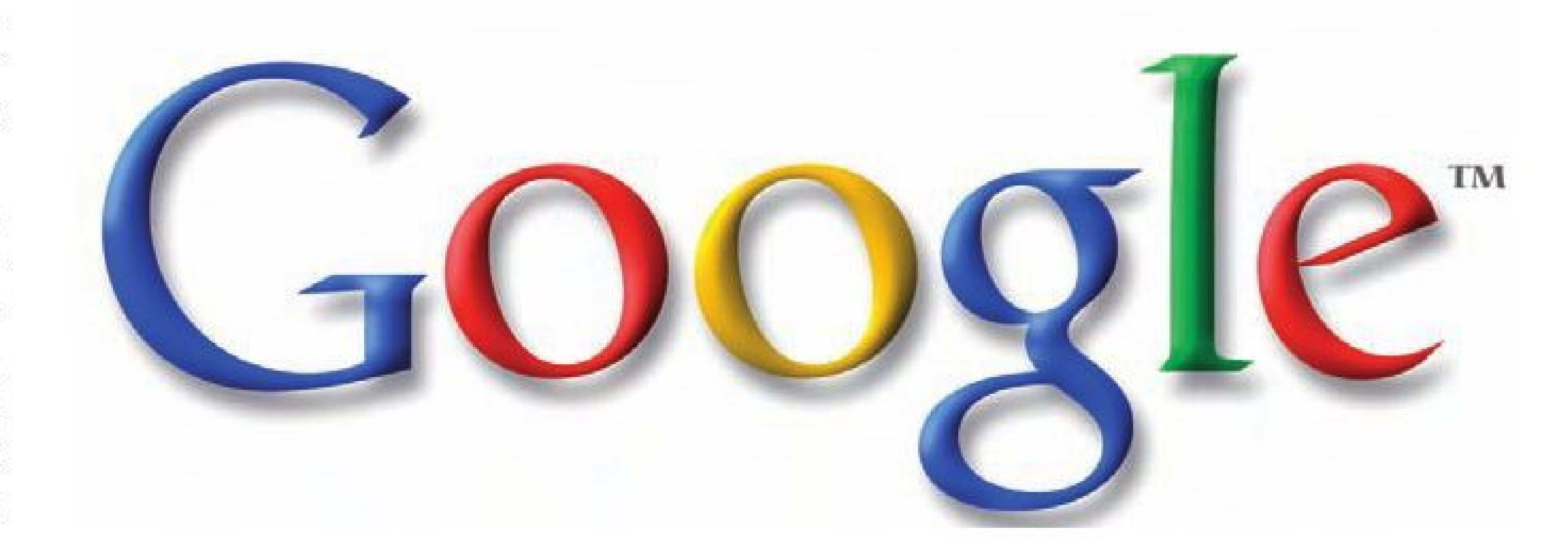
[1] http://h10010.www1.hp.com/wwpc/us/en/sm/WF06b/321957-321957-64295-321838-306995-3687084-3687085-3710156.html?jumpid=oc_R1002_USENC-001_HP%202133%20Mini-Note%20PC&lang=en&cc=us
[2] http://h71016.www7.hp.com/dstore/ctoBases.asp?BEID=19701&ProductLineId=539&oi=E9CED&FamilyId=2769&LowPrice=%24729.00&LowBaseId=23430&jumpid=reg_R1002_USEN#

Más información:
Nota de prensa de HP
http://www.hp.com/hpinfo/newsroom/press/2008/080408xc.html

Google proporciona infraestructuras a los programadores de aplicaciones web

La compañía del buscador pone a disposición de la comunidad de desarrolladores de software toda la infraestructura necesaria para que puedan alojar sus aplicaciones web 2.0 y ofrecerlas a los usuarios. Google App Engine[1] es la nueva apuesta de Google para los programadores, una comunidad a la que la compañía del buscador trata con sumo cariño, ofreciéndole todo tipo de recursos, facilidades y API's de sus propios productos para que las utilicen a discreción.
El App Engine de Google no es más que una modalidad de hosting que proporciona al programador no solamente un espacio físico en disco para depositar sus ficheros de manera que se encuentren siempre accesibles a los internautas, sino también recursos para la ejecución de la aplicación, cómo ancho de banda y memoria.
Estos recursos son escalables, con lo que si la aplicación diseñada necesita más, la infraestructura de servidores de Google le proporcionará lo que necesite.
Los programadores también podrán echar mano de las API's de los productos de la compañía del buscador, lo que les ahorrará tiempo y esfuerzos a la hora de implementar y utilizar sistemas cómo la validación del usuario, así cómo poder interactuar desde la aplicación creada con otras aplicaciones web cómo son Gmail o Google Docs.
Por el momento se ha abierto el hosting en versión preview limitada a los primeros 10.000 desarrolladores que se inscriban, con un máximo de 500 MB. para ficheros y el ancho de banda para servir cinco millones de páginas mensuales. Para el futuro parece que continuará habiendo una versión gratuita limitada y los programadores interesados podrán adquirir una mayor cantidad de recursos en los servidores de Google a medida que los vayan necesitando.

Más información:
Nota de prensa de Google
http://www.google.com/intl/en/press/annc/20080407_app_engine.html
[1] http://code.google.com/appengine/





Así podría ser Windows 7

La interfaz multitáctil y la posible inclusión del sistema de ficheros WinFS, principales novedades que -haciendo un poco de futurología- podría incluir la próxima versión de Windows.

Parece que Microsoft no quiere que la próxima versión de su sistema operativo tarde tanto tiempo en aparecer en el mercado cómo Vista, que ha salido seis años después de Windows XP.

De hecho, en sus cuarteles generales de Redmond se trabaja siempre en las dos siguientes versiones del sistema operativo, por lo que los programadores, testeadores y directivos se encuentran varios años por delante del resto de los mortales en cuanto a tecnología informática se refiere.

Por ahora solamente disponemos de rumores sobre las novedades que supuestamente aportará ésta nueva versión de Windows, pues ninguna información oficial ha salido de Microsoft.

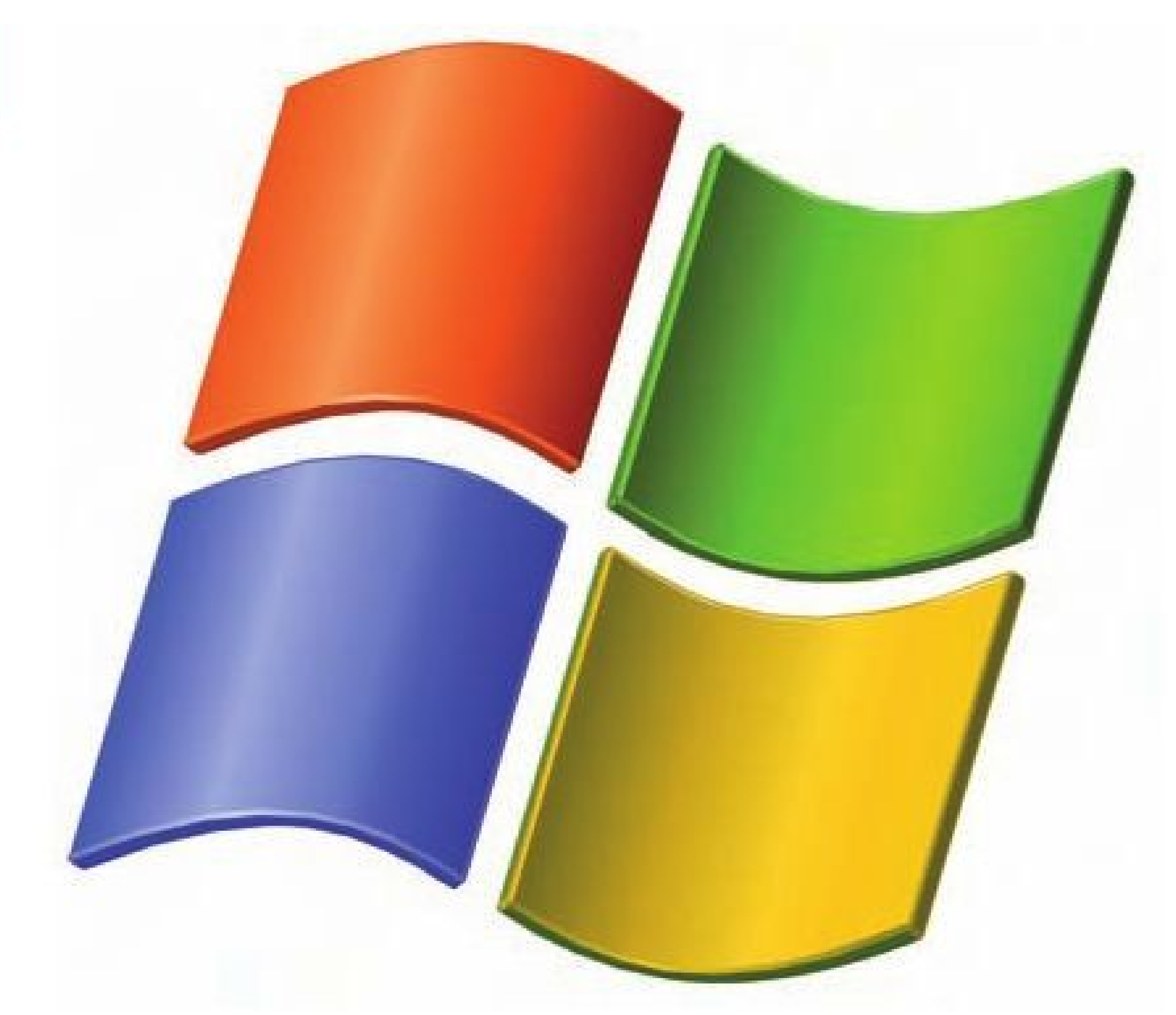
Interfaz multitáctil y reconocimiento de voz

Es la gran moda en el sector de la telefonía móvil desde que Apple la introdujo en el iPhone y posteriormente en el iPod Touch, y consiste en una interfaz táctil que en lugar de capturar solamente la presión realizada en un punto por un lápiz, es capaz de reconocer y administrar su uso con el dedo en movimientos que no son solamente una pulsación puntual, sino arrastrándolo sobre la pantalla táctil del dispositivo.

En comparación con la superficie de toque del tradicional puntero incluido en éste tipo de dispositivos, el dedo presenta una mayor superficie, por lo que también requiere un número mayor de cálculos para precisar la operación que el usuario quiere realizar.

La interfaz multitáctil ha desbordado a Apple, siendo adoptada por otros fabricantes cómo HTC, que la incluye en forma de software instalado sobre el sistema Windows Mobile en el HTC Touch por ejemplo, y es probable que se convierta en otra tecnología de futuro adoptada por todos los fabricantes pero introducida inicialmente por los muchachos de Steve Jobs.

Windows 7 podría ser el primer sistema operativo en introducir esta funcionalidad para computadoras de sobremesa y portátiles, indicada especialmente para máquinas de tipo Tablet PC (de las que ahora se está viendo un "revival") y UMPC. Así, es posible que para introducir datos en un Tablet PC o dispositivo similar ya no necesitemos de un puntero especial, sino que en el futuro podamos hacerlo solamente con un dedo.



La tecnología de reconocimiento de voz que permite darle órdenes al ordenador sin tener que tocar el teclado, o dictarle los textos a enviar por correo electrónico o los documentos y que ya encontramos en Vista, seguirá en Windows 7 pero corregida y aumentada.

WinFS

El sistema de ficheros que debía ser el sucesor del NTFS en Windows Vista finalmente no vio la luz con éste sistema operativo, aunque el proyecto todavía sigue su desarrollo. Su concepción reposa sobre la misma idea de las bases de datos relacionales, y teóricamente debe ser mucho más eficiente con grandes volúmenes de almacenamiento cómo los discos duros que se están vendiendo actualmente. También deberá facilitar la tarea de las herramientas de indexación y búsqueda de escritorio, cada vez más populares y necesarias.

Es posible que veamos a WinFS incluido en Windows 7, aunque esto siempre dependerá de que el proyecto se finalice a tiempo, algo que puede parecer factible debido al tiempo que hace que se inició pero que también podría verse en peligro por el corto margen disponible hasta la fecha de lanzamiento que Microsoft quiere dar a Windows 7.

Un elemento que se vería muy ayudado por WinFS sería la llamada "búsqueda semántica" en nuestro sistema, que consiste en buscar no por palabras clave, sino por expresiones naturales. Así, actualmente si queremos saber quien descubrió América mediante el uso de un buscador de Internet, pondríamos en él los términos "descubrimiento América", pero con la búsqueda semántico introduciríamos la pregunta cómo la haríamos a otra persona: "¿quién descubrió América?".

Estos sistemas ya se están experimentando en Internet con éxito diverso, pero las necesidades a medio-largo plazo y la progresión de la tecnología permitirán que podamos implementar estas herramientas en nuestras máquinas desktop.

MinWin

El sistema será más modularizable, y lo que más destacará será el núcleo (kernel) que por el momento recibe el nombre de MinWin. Ocupa solamente unos 25 MB. de espacio en disco y usa unos 40 de memoria. Será la base para poder construir un sistema a medida, algo que ya vemos con las diferentes versiones de Windows Server 2008 en que si bien la modularidad se hace más "a groso modo" con los servicios del sistema, la idea es la misma.

Horizonte 2009 a la vista

Al principio de éste artículo me he referido a que Microsoft no quiere que esta nueva versión salga al mercado con tanta diferencia de tiempo respecto a la anterior (el actual Windows Vista) cómo esta lo hizo de Windows XP (seis años en total).

La fecha barajada es algún momento de 2009, probablemente durante la segunda mitad, lo que daría un lapso de a penas dos años respecto al lanzamiento comercial de Vista. Y desde las capas altas de la compañía se quiere que a partir de ahora esta sea más o menos la ventana de lanzamiento de una nueva versión del sistema operativo para poder combatir la asiduidad con la que se lanzan nuevas versiones de Linux y de Mac OS X, sus principales rivales en el sector desktop y, en el primer caso, también en los servidores.

Naturalmente, todas las funcionalidades comentadas aquí son solamente rumores más o menos fundamentados que se han ido filtrando a Internet, a veces de fuentes no reveladas y otras en declaraciones realizadas por altos cargos de Microsoft, cómo los propios Bill Gates o Steve Ballmer.

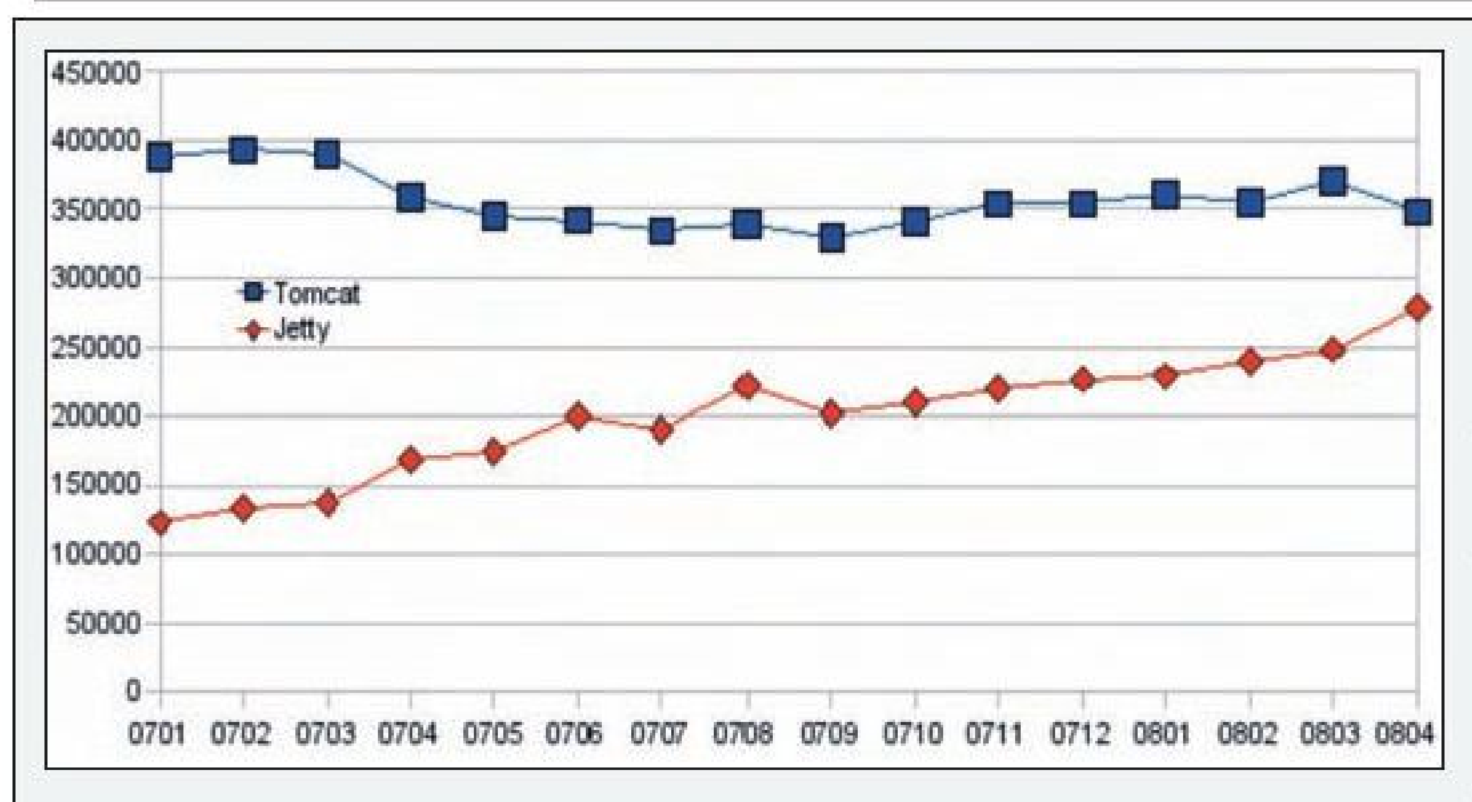
Autor: Guillem Alsina
guillem@imatica.org





Actualidad Java de la mano de javaHispano

Jetty podría superar a Tomcat en número de despliegues a finales de año

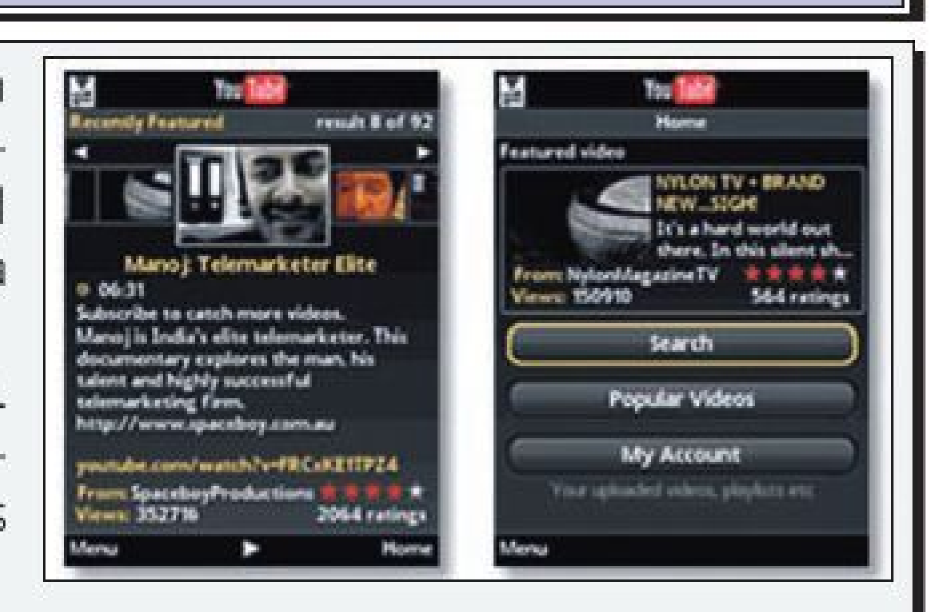


Durante bastante tiempo Tomcat ha sido el servidor web Java EE más popular con diferencia. Sin embargo, según las estadísticas de Netcraft, en el último año y medio Jetty ha experimentado un considerable crecimiento: actualmente existen ocho despliegues de Jetty por cada 10 Tomcats, y en los últimos seis meses Jetty ha ganado 9000 despliegues nuevos por mes. De seguir a este ritmo, antes de terminar el año podría superar el número de despliegues de Tomcat, ya que (como puede verse en la imagen) este servidor apenas ha presentado una variación en el número de instancias desplegadas en el último año y medio.

Entre los puntos fuertes que podrían estar apoyando el crecimiento de Jetty están lo ligero del servidor web, su rapidez y su sencillez de configuración frente a Tomcat. Jetty se distribuye bajo licencia Apache 2.0 y es desarrollado por la compañía Webtide, que vende servicios en torno a él. Recientemente Eclipse ha decidido distribuirlo integrado con el entorno de desarrollo, y Jetty es el contenedor de Servlets empleado por JBoss y Apache Gerónimo.

Reproductor de videos de Youtube basado en Java ME

Youtube ha anunciado un reproductor de videos, por lo de ahora sólo en versión beta, basado en el perfil MIDP2 de Java ME. El reproductor ofrece a los usuarios una experiencia más rica e interactiva que la que se obtiene actualmente mediante la versión web para terminales móviles del portal. Por lo de ahora los terminales soportados son: los N73, E65, N95, 6120c y 6110n de Nokia y los k800i y w880i de Sony Ericsson.
El reproductor puede descargarse, de un modo totalmente gratuito, desde m.youtube.com/app. Existen dos versiones de la aplicación, una firmada que tendrá permisos totales de acceso a nuestro terminal móvil, y otra sin firmar que solicitará permiso al usuario antes de iniciar acciones potencialmente peligrosas, como descargarse contenido de la red.



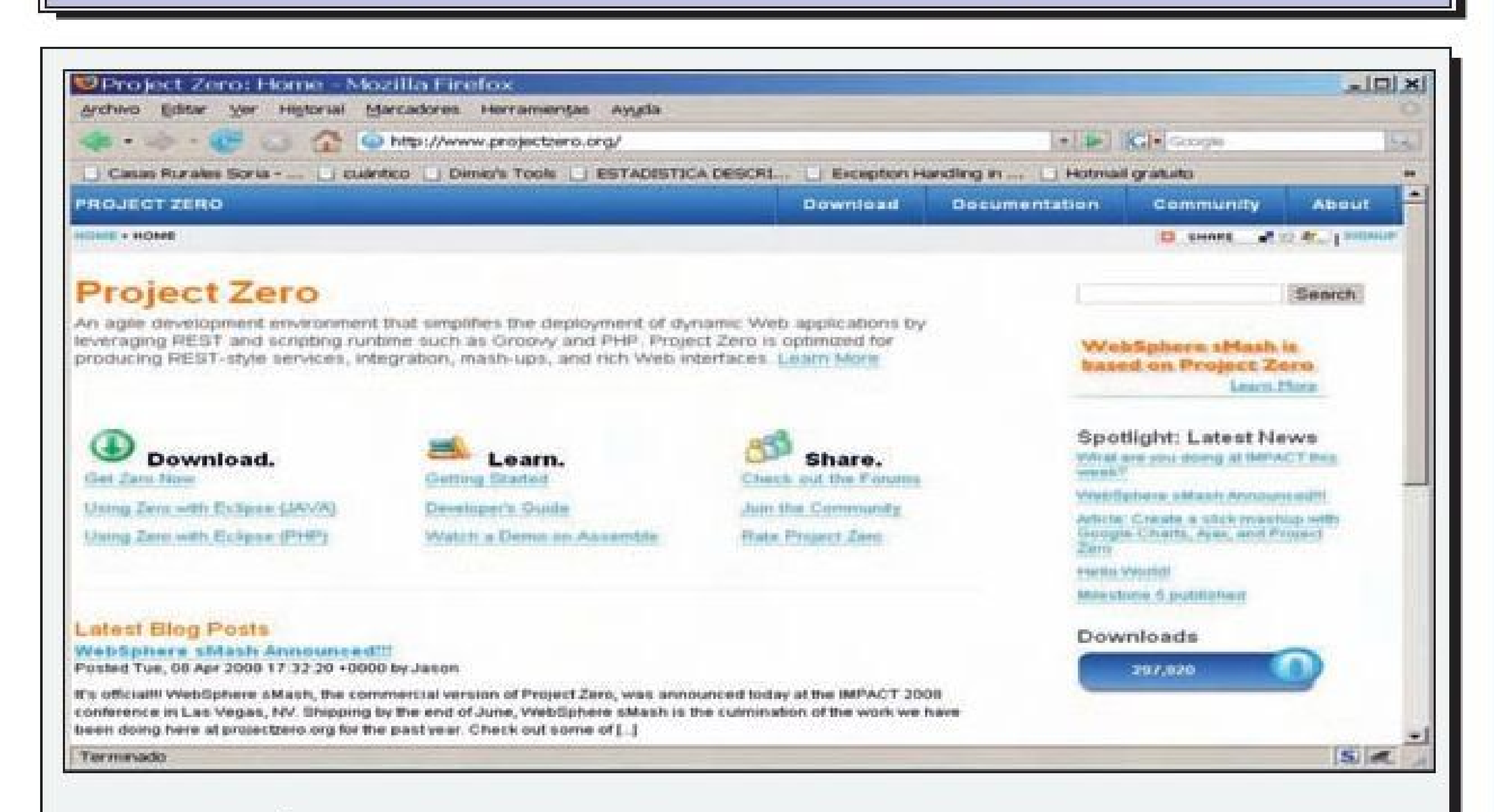


Code Gear anuncia JBuilder 2008

Code Gear, la filial de Borland que ahora lleva todo su negocio relacionado con entornos de desarrollo, ha anunciado la segunda versión de JBuilder desarrollada sobre Eclipse: JBuilder 2008. Las principales novedades son la actualización de los servidores de aplicaciones que soporta, la inclusión de un profiler (antes la compañía ya lo tenía, pero ahora viene incluido con el IDE), mejoras en el análisis estático y métricas de código, mejoras en el soporte para colaboración, y una nueva versión de InterBase.
Además, JBuilder 2008 proporciona un mejor soporte para importar proyectos creados con versiones de JBuilder anteriores a 2007, es decir, anteriores al momento en que comenzó a basarse en Eclipse. Todavía está por ver si JBuilder 2008 y sus versiones posteriores volverán a ser el producto estrella que fue a principios de esta década, momento en el que era líder indiscutible de los entornos de desarrollo Java. Lo que sí es un hecho es que su antecesor, JBuilder 2007, ha tenido una existencia, cuanto menos, discreta y a la sombra de Netbeans, Eclipse e IntelliJ.



IBM anuncia Project Zero, un stack al estilo Ruby on Rails



Project Zero (su nombre viene de "Zero complexity. Zero overhead. Zero obstacles"), también llamado "WebSphere sMash", es un stack para el desarrollo de aplicaciones web con una filosofía muy similar a la de Ruby on Rails, que puede usar como lenguajes de programación Groovy, PHP o Java. Este framework, creado por IBM, está orientado a la creación de servicios tipo REST (internamente emplea Dojo para exponer dichos servicios), mash-ups y, en general, el desarrollo de aplicaciones tipo "web 2.0".

Si bien tecnológicamente se trata de un producto muy interesante, su licencia es un tanto controvertida. Por un lado, en un momento en el que prácticamente todos los framework web son software libre han optado por una licencia propietaria. Esta licencia impone varias limitaciones orientadas a impedir que proyectos con un volumen grande de usuarios tomen ventaja de esta solución. Por ejemplo, no se puede ejecutar en más de cuatro cores de modo simultáneo y no se pueden tener más de cuatro instancias en un mismo CPD. Por lo de ahora IBM tampoco ofrece soporte comercial o licencias para el proyecto; cualquier uso que exceda los límites impuestos por la licencia deberá ser negociado individualmente con la compañía.

Sun ofrece soporte extendido para Java SE para negocios



Sun ha anunciado un nuevo servicio de soporte extendido para su implementación de Java SE. El servicio está orientado a grandes empresas y consiste en que Sun seguirá proporcionando soporte, parches de seguridad y correcciones de bugs, e incluso aceptando peticiones concretas de los clientes para solucionar problemas en su JDK y JRE durante más del doble del tiempo de soporte habitual: hasta 15 años.

Este servicio puede contratarse para Java 1.4.2, 5 y 6; para versiones anteriores no está disponible. Su precio comienza en 10 $ por empleado y año para el nivel de soporte más básico, y 12.5 $ por empleado y año para el nivel de soporte premium. El servicio está orientado a compañías conservadoras que realizan cambios en su infraestructura de un modo muy lento, como pueden ser bancos o grandes financieras.

OPINIÓN

To closure or not to closure

Un tema candente actualmente en el mundo Java es la introducción, o no, en Java 7 de otro elemento de programación: las "closures". Simplificando, muchos lenguajes permiten manejar bloques de código parametrizable como si fueran datos, permitiendo asignarlos a variables, pasarlos como parámetro etc. La solución actual de Java a este problema, las "inner classes", deja insatisfecha a mucha gente debido a sus limitaciones y poca elegancia comparada con otros lenguajes.

Sin embargo, al mismo tiempo que ha crecido el apoyo para añadir "closures" a Java, también ha crecido el movimiento opuesto, el cual argumenta, entre otras cosas, que es demasiado tarde para un cambio radical en el lenguaje



y que las especiales características de Java recomiendan cautela. El debate subyacente, el caso de las "closures" es sólo una de las batallas, enfrenta por un lado a los desarrolladores que prefieren ver como Java evoluciona y "se pone al día" siguiendo la estela de otros lenguajes, y los que prefieren priorizar el "espíritu Java" de lenguaje consolidado, simple, fácil de aprender, con compatibilidad asegurada, y dejar los aspectos más "dinámicos y modernos" para otros lenguajes, con los que se puede interactuar, si hace falta.

El debate promete ser interesante y aunque la implementación final dejará a mucha gente insatisfecha en cualquier caso, la discusión, alimentada por la recientemente estrenada naturaleza Open Source de Java, destila libertad y será, en todo caso, enriquecedora.

Daniel Lopez Janariz, Community Manager de la JavaTools Community de java.net

Sobre el autor

Abraham Otero (abraham.otero@javahispano.org) es responsable de calidad y miembro de la junta de javaHispano.



Un fenómeno llamado Google Bombing

NICOLÁS VELÁSQUEZ ESPINEL

Llevamos algunos años oyendo de un tipo de "ataque social" denominado Google Bombing mediante el cual, a base de asociar múltiples veces un término o texto con un enlace en una web, se consigue llevar dicho enlace a las primeras posiciones de los resultados de búsqueda de Google. Entre sus víctimas más ilustres Microsoft, George Bush o, más recientemente, el polémico y popular caso SGAE=ladrones.

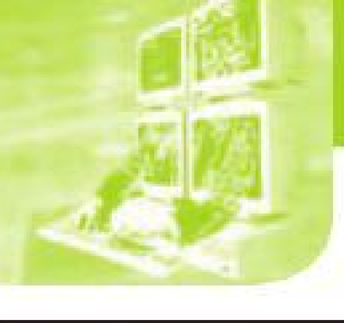
Sin ninguna duda Google lleva varios años liderando el mercado de los buscadores en Internet, destacando entre otras cosas por su velocidad y el acierto en la mayoría de los casos gracias a la conjunción de varios factores como un algoritmo de búsqueda muy efectivo, a una herramienta austera y a su sistema de posicionamiento bautizado como Page Rank.
Gracias a esta poción mezcla de ingenio, eficiencia y sencillez, millones de usuarios y empresas, que antes contaban muy poco de cara a los resultados de una búsqueda, comienzan a aparecer en lugares relevantes, al fin y al cabo el secreto del éxito en la red. Y es que estadísticamente si deseas llegar a muchos usuarios y sólo apareces en la página 25 de resultados, olvídate de que este medio pueda suponer un elemento esencial a modo promocional.
Situar una empresa en los primeros lugares de una búsqueda no es una tarea sencilla. Hace algunos años bastaba con darse de alta en algunos buscadores pero hoy en día el éxito de un buen posicionamiento requiere mucho más (hay empresas que se dedican exclusivamente a esto), incluida la popularidad y, porqué no, en algunos casos un elemento monetario de por medio. Pero de un tiempo para acá apareció, mezcla de un sentimiento de rebeldía social unida una dosis de picaresca tecnológica, un sistema mediante el cual era posible asociar determinados términos, normalmente reivindicativos, a una empresa o entidad, de manera que aparecieran directamente relacionadas al realizar una búsqueda, siempre aprovechando el método que Google empleaba para llevar a cabo su sistema de posicionamiento. En definitiva un sistema reivindicativo poco ortodoxo pero que ha llegado a ser muy popular y que fue bautizado en su momento como Google Bombing.

La técnica

De manera resumida se trata de conseguir que una determinada página web aparezca en la primera posición de los resultados de Google como resultado de la búsqueda de una o varias palabras específicas. Es en definitiva la misma técnica empleada para el posicionamiento web, pero llevada al terreno de la reivindicación social o política como su telón de fondo gracias a un tratamiento "especial" de ciertos parámetros.
Y es que el método es sorprendentemente sencillo. Todo arranca de una iniciativa propuesta por un promotor que solicita a la comunidad internauta que desee participar (que pueden ser responsables de otro sitio webs, bloggers, etc.) que inserten en sus páginas un código específico. Información transparente para el navegante, sencilla de introducir por el webmaster y diáfana para el robot de Google que recorre la red catalogando sus contenidos. Si esta propuesta gana adeptos y la adoptan miles de internautas solidarios con la causa inicial, se conseguirá el efecto deseado: la aparición en los primeros lugares de una búsqueda teóricamente cándida.
En concreto la técnica consiste en incrustar en el código de la página un enlace de la forma:

```
<A href="http://url_de_la_pagina">palabra1
palabra2</A>
```

Este acto en sí mismo no dejaría de ser una iniciativa totalmente inocua si no fuera por que se aprovecha del pilar sobre el que está basado el potente buscador, los enlaces, utilizando en su beneficio el algoritmo de clasificación de Google, el PageRank.
Pero para ello es necesario cómo funciona este algoritmo, por lo menos a grandes rasgos. El sistema sigue la siguiente pauta: cuantos más enlaces obtenga una determinada página con una palabra (o términos) en particular, más posibilidades tendrá de aflorar en las primeras posiciones. A "grosso modo" se trata de un sistema que premia a las páginas más populares (que no más visitadas, aunque muchas veces vayan de la mano), es decir, las más mencionadas y enlazadas de la red. En definitiva, cuanto más se hable de ti, mejores resultados obtendrás en el ranking de búsquedas (hasta pueda darse el caso






Seomoz propone la manipulación de la Wikipedia para "perjudicar" a otros.



El Blog Merodeando inicio la polémica al publicar una entrada con el título SGAE=Ladrones.

paradójico de que dentro de la página web objetivo del Google Bombing no aparezca ni una sola vez las palabras de la búsqueda).

Visto que requiere de la colaboración de miles de personas que persigan una misma causa, es muy poco probable que alcance el éxito con una iniciativa de carácter privado, lo que explica por qué el impacto mediático conseguido hasta ahora ha tenido como temática principal determinados asuntos de índole social, normalmente por asuntos de carácter político o popular que adquieren una relevancia suficiente entre la comunidad internauta.

El objetivo es bien claro: asociar e identificar a la página web víctima del Google Bombing con una palabra o términos dados. Y para muestra un botón: con toda probabilidad uno de los casos más populares a nivel mundial (y que además nos toca directamente) ha sido el fenómeno de la "SGAE=ladrones", un acontecimiento que levantó ampollas entre la entidad española de gestión de derechos de autor ya que como efecto directo del Google Bombing comenzó a aparecer en primer lugar de la lista de resultados del famoso buscador cuando se buscada el término "ladrones". En poco tiempo el asunto llegó a las primeras planas de los medios de comunicación.

Para evitar males mayores, Google tomó cartas en el asunto a instancias del organismo español, como lo ha tenido que ir haciendo con otras iniciativas similares llevadas a cabo en otros países, ya que dispone de los mecanismos para evitar este tipo de situaciones desde hace tiempo, aunque no hubiera hecho uso de estos hasta entonces (de hecho, lo demostró claramente en noviembre de 2003 cuando implementó un filtro para combatir el spam).

El secreto está en el PageRank

El éxito de Google y, por lo que se ve, también su punto débil (visto el éxito del Google Bombing) tiene nombre y apellido: PageRank. El algoritmo de PageRank fue patentado en los Estados Unidos el día 8 de enero de 1998, por el propio Larry Page. El nombre técnico que recibió entonces fue "Method for node ranking in a linked database", y le fue asignado el número de patente 6,285,999.

PageRank™ es sencillamente un valor numérico que representa la importancia que una página web tiene en Internet. Ahora el truco está en saber cómo se asigna dicho valor. Para ello Google se hace la idea de que cuando una página asigna un enlace a otra, es de hecho un voto para esta última (si yo pongo un enlace a alguna página es porque, de alguna forma, la recomiendo). Cuantos más votos tenga una página, más importante será ésta para Google. Como valor añadido, la importancia de la página que emite su voto también determina el peso de éste. De esta manera, Google calcula la importancia de una página gracias a todos los votos que reciba, teniendo en cuenta también la importancia de cada página que emite el mismo.

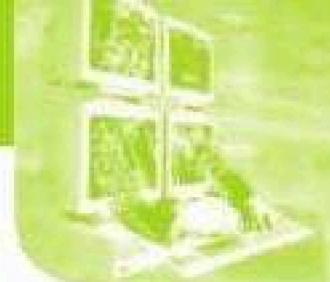
PageRank (desarrollado originalmente por los fundadores de Google Larry Page y Sergey Brin) ofrece de esta manera a Google una forma de decidir la importancia de una página en concreto. Se trata de un dato valioso (no es el único pero sí es uno de los más importantes), porque es uno de los factores que determinan la posición que va a tener una página dentro de los resultados de la búsqueda.

Sin embargo, hay que tener en cuenta que no todos los enlaces son tenidos en cuenta por Google. Por de pronto, Google filtra y descarta los enlaces de aquellas páginas dedicadas exclusivamente a situar enlaces (son las llamadas "link farms" o granjas de enlaces).

Además, Google es consciente de que una página no puede controlar los enlaces que apuntan hacia ella, aunque sí que puede hacerlo con los enlaces que esta web coloca hacia otras páginas. Por ello, los enlaces que una página coloque hacia sitios que puedan considerarse penalizados, pueden llegar a ser perjudiciales para su propio valor de PageRank. Si un sitio web tiene PR0 (PageRank 0), generalmente se tratará de una web penalizada, y podría ser poco inteligente colocar un enlace hacia ella. Para ello si queremos cuidar nuestro PageRank tendremos que conocer previamente el de las páginas hacia la que emplacemos enlaces en nuestra página. Una forma de poder hacerlo es descargándonos la barra de búsqueda de Google (solamente disponible para Internet Explorer). Esta muestra una barra en la que se muestra en color verde el valor de PageRank de la página que estemos visitando en una escala de 0 a 10. Por ejemplo sitios web con PR10 son Yahoo!, Microsoft, Adobe, o la propia Google, como no.

Microsoft: el primer objetivo

El caso de Microsoft en el 2002 es probablemente el primer caso reconocido en donde se implementó este tipo de ataque, si puede denominársele así. En aquel entonces, cuando se buscaba las palabras "go to hell" ("vete al infierno", en inglés) aparecía como primer resultado la página de Microsoft, una empresa que, como es bien sabido, genera cierta animadversión en un gran sector de la población internauta (también se habla de un ataque anterior a George



Google

An explanation of our search results.

If you recently used Google to search for the word "Jew," you may have seen results that were very distubing. We assure you that the views expressed by the sites in your results are not in any way endorsed by Google. We'd like to explain why you're seeing these results when you conduct this search.

A site's ranking in Google's search results relies heavily on computer algorithms using thousands of factors to calculate a page's relevance to a given query. Sometimes subtleties of language cause anomalies to appear that cannot be predicted. A search for "Jew" brings up one such unexpected result.

If you use Google to search for "Judaism," "Jewish" or "Jewish people," the results are informative and relevant. So why is a search for "Jew" different? One reason is that the word "Jew" is often used in an anti-Semitic context. Jewish organizations are more likely to use the word "Jewish" when talking about members of their faith. The word has become somewhat charged linguistically, as noted on websites devoted to Jewish topics such as these:

- http://shakti.trincoll.edu/~mendele/vol01/vol01.174
- http://www.jewishworldreview.com/cols/jonah081500.asp

Someone searching for information on Jewish people would be more likely to enter terms like "Judaism," "Jewish people," or "Jews" than the single word "Jew." In fact, prior to this incident, the word "Jew" only appeared about once in every 10 million search queries. Now it's likely that the great majority of searches on Google for "Jew" are by people who have heard about this issue and want to see the results for themselves.

The beliefs and preferences of those who work at Google, as well as the opinions of the general public, do not determine or impact our search results. Individual citizens and public interest groups do periodically urge us to remove particular links or otherwise adjust search results. Although Google reserves the right to address such requests individually, Google views the comprehensiveness of our search results as an extremely important priority. Accordingly, we do not remove a page from our search results simply because its content is unpopular or because we receive complaints concerning it. We will, however, remove pages from our results if we believe the page (or its site) violates our Webmaster Guidelines, if we believe we are required to do so by law, or at the request of the webmaster who is responsible for the page.

We apologize for the upsetting nature of the experience you had using Google and appreciate your taking the time to inform us about it.

Sincerely,
The Google Team

p.s. You may be interested in some additional information the Anti-Defamation League has posted about this issue at http://www.adl.org/rumors/google_search_rumors.asp. In addition, we call your attention to Google's search results on this topic.

©2007 Google - Google Home - About Google

Comunicado de Google acerca del resultado de la búsqueda de la palabra "Jew".

Bush en el 2000 asociándole al término "dumb mother fucker" aunque en este caso lo que se hacía era enviarnos a una página que hablaba del presidente de los EEUU).

Cuando se destapó el caso de Microsoft, muchos fueron los que se preguntaron cómo había podido hacer para superar en el ranking a la propia página Hell.com (infierno.com) para aparecer en la primera posición de las búsquedas bajo la categoría "go to hell". Algunos sugirieron que podría deberse al método que empleaba Google para conseguir resultados por medio de su "análisis de enlaces" y acertaron de pleno. El buscador mostraba los resultados, no sólo en base a las webs que contenían esa palabra (como se hacía hasta entonces), si no que también lo hacía en función de otras que estaban enlazadas a la palabra o frase. Esto significaba que ya ni si quiera era necesario que la página en cuestión tuviera que contener la frase "go to hell" para aparecer referenciada a este término en el buscador.

Sin embargo, aunque si fue el primero, Microsoft no fue el único que sufrió las secuelas de esta nueva tendencia ya que otras webs como la de America Online y la de Walt Disney también aparecieron situadas dentro de los cinco primeros resultados de la misma búsqueda. Nacía entonces el Google Bombing, un sistema de rebeldía virtual social y se marcaba el inicio de una época en la que muchos fueron conscientes de lo sencillo que podía llegar a ser "engañar" a Google.

Poco después, el 27 de noviembre del 2003, el weblog www.blah3.com/graymatter/ (ya no está operativo) propuso un Google Bombing contra el presidente norteamericano George Bush. Allí se citaba textualmente: "A partir de este día, me referiré a George W. Bush como un Miserable Fracaso al menos una vez al día".

El autor de la iniciativa pretendía con esto incluir un enlace hacia la web de la Casa Blanca con el texto "Miserable Failure" ('Miserable Fracaso') y animaba a otros usuarios a hacer lo propio. Al poco tiempo había conseguido su propósito. Si buscábamos en Google las palabras "miserable failure", aparecía en primer lugar la web indicada aunque en ningún caso este texto apareciera como parte del contenido de la página web de la Casa Blanca. Como dato curioso, cabe destacar que la frase "George Bush es un miserable fracaso en política exterior" fue pronunciada unas semanas antes por Dick Gephardt, candidato demócrata a la Presidencia del Gobierno norteamericana en las elecciones que se iban a celebrar en el año 2004. Incluso Gephardt tenía registrado el dominio "amiserablefailure.com".

Ya en abril del mismo año se destapó un caso que tocó la sensibilidad de un gran sector de la población. Cuando se buscaba en Google la palabra judío en inglés (jew), aparecía situada en primera posición la página de "'jewwatch.com", un sitio web considerado como "antisemita". Tal fue el revuelo que se montó debido a este ataque de Google Bombing (la comunidad judía en los Estados Unidos es especialmente poderosa) que la empresa norteamericana responsable del buscador - y cuyo cofundador y presidente es judío - tuvo que hacer público un comunicado, que enlazó además desde la página de resultados en forma de AdWords (publicidad integrada en las búsquedas).

El Google Bombing ha saltado a mundo real a través de los billetes de euros.

Ese mismo año el fenómeno comenzó a extenderse por el resto del mundo y, al igual que le ocurriera al presidente de EEUU, al de Ecuador y al de Dinamarca, fue utilizado por los detractores del presidente de España, José Luis Rodríguez Zapatero, aprovecharon la popularidad del buscador Google para lanzar una campaña contra él e intentar desprestigiarle. De esta forma, gracias a la rápida propagación de la iniciativa a través de diversos blogs en español, un nuevo Google Bombing se puso en marcha, y el sitio web oficial de la campaña de Zapatero (http://www.zapateropresidente.com/) aparecía en 2004 en los primeros lugares al realiza la búsqueda de la palabra "gafe" en Google. En la actualidad la palabra "gafe" y "Zapatero" sigue en primer lugar aunque el enlace dirige hacia otra página web.

La realidad es que pocos son los casos en los que los afectados han tomado medidas contra aquellos que defendían o impulsaban el 'Google Bombing', quizás conscientes que ello conllevaría un flagrante ataque al derecho de la libre información sobre el que se asientan las bases de la propia Internet y que, al fin y al cabo, supondría de alguna manera, un intento por acallar una crítica social y, en definitiva, una opinión pública. Sin embargo España se erigió como protagonista en este aspecto cuando la SGAE denunció a un blogger con el objetivo de eliminar una página web de los resultados del buscador web de Google, caso que trataremos más adelante.

Otro suceso a destacar es el del ciudadano polaco de 23 años Marek W. que fue detenido por las autoridades de su país por "insultar al presidente Lech Kaczynski" (según cuenta Philipp Lenssen en su blog). Este joven consiguió que la página oficial del Primer Ministro apareciese en la primera posición del buscador web de Google al consultar la palabra "kutas" ("pene", en su alocución más políticamente correcta), tras lo cual la Policía polaca consiguió dar con él rastreando su dirección IP (lo había hecho desde el ordenador de su casa). El detenido ha asegurado que simplemente trataba de demostrar sus habilidades de programación informática desarrollando una herramienta para situar un sitio web en la primera posición de Google cuando se busca una palabra (una aplicación SEO). Actualmente piden 3 años de cárcel.

Uno de los casos más recientes ha sido el que ha protagonizado El Corte Inglés que por lo visto ha sido víctima de un caso de Google Bombing. En este caso la acción se ha visto realizada desde una vertiente sindical, UGT, que después de llevar varias veces a los tribu-



nales a esta empresa por su política laboral y ver que no conseguía en ningún caso el interés de los medios de comunicación, decidió llevar a cabo el Google Bombing correspondiente. Esto ha provocado que se hayan podido encontrar diversas páginas con denuncias sobre la forma de llevar a cabo la política laboral de su empresa. Desde puestos de trabajo con riesgos para la salud, incumplimiento de descansos semanales y calendario laboral, discriminación salarial y hasta el curioso caso en el que obtienen beneficios de los propios salarios de los trabajadores ingresando las nóminas en una entidad financiera de la propia empresa.

Otros ilustres casos de Google Bombing que han logrado obtener cierta repercusión son:

- Worst president - George Bush
- Great president - George Bush (también¿?)
- Petrolero Prestige
- John Kerry – "waffles"
- 'jew' ('judío')
- Jacques Chirac - "magouilleur"
- Jan Peter Balkenende - 'raar kapsel'
- El príncipe holandés Willem-Alexander para la búsqueda slechte tanden ("dientes malos"), debido a su mal cuidada dentadura.
- Microsoft Internet Explorer – "insecure"
- Bastards - SCO group
- Leave now – Disney

El caso SGAE

El viernes 23 de abril de 2004 a las 01:26, Julio Alonso posteaba en su blog "Merodeando" lo que ha sido posiblemente uno de los mayores de cabeza para la SGAE en los últimos años y, con toda seguridad, el caso más popular de Google Bombing en territorio español. El título del mismo dejaba claro el contenido tratado: "SGAE = ladrones". Al poco tiempo, al escribir la palabra "ladrones" en el buscador como término de búsqueda, aparecía como primer resultado la página web de la Sociedad General Española de Autores.

La SGAE, ni corta ni perezosa, atacó directamente al mensajero, a Google, acusando incluso al buscador de "fascista" (según palabras de su presidente, Teddy Bautista, que llegó a comparar este ataque con el problema de la pornografía infantil) y haciéndole directamente responsable - y de forma voluntaria - de que aparecieran dichas referencias despectivas contra la Asociación, desvirtuando de un plumazo el poder de la masa social que ha venido criticando de forma continuada su comportamiento.

El desconocimiento y el mal asesoramiento fueron seguramente los responsables de las desafortunadas declaraciones del señor Bautista (que, todo hay que decir, fueron todo menos acertadas), al igual que lo fueron las vertidas por el diario vasco "El Correo" que en un principio aseguraba que había sido la propia Google quien había organizado el ataque de Google Bombing contra la SGAE.

Hoy en día ya se sabe que esta campaña tuvo como origen la multitud de usuarios españoles que se muestran en contra del pago de un canon a la SGAE cada vez que se compra un soporte de almacenamiento digital. Un ataque de esta magnitud sería impensable de otra forma. Y es que poco tiempo después del post de Julio Alonso, cientos, quizás miles, de páginas se hacían eco de este titular en lo que se ha convertido por méritos propios en el ataque de Google Bombing más importante de nuestro país. Hoy en día incluso puede verse desde el post cómo ha sido la denuncia judicial.

Para deshacer el entuerto, y eliminar el resultado de las primeras posiciones de la citada búsqueda, la SGAE empleó la vía judicial con el fin de modificar los resultados del buscador web de Google, a sabiendas que la empresa norteamericana elimina automáticamente los enlaces a cualquier sitio web una vez que existe alguna demanda judicial que la respalda. A esto hay que unir que a Google no le hace ninguna gracia los ataques de Google Bombings ya que, en definitiva, se trata de una técnica que no hace más que aprovecharse del diseño de su algoritmo de clasificación de relevancia del buscador para posicionar diversos sitios web al buscar ciertos términos. Algo así como su talón de Aquiles.

Sin embargo hay que destacar que esta iniciativa no evitará que sigan apareciendo resultados "incómodos" para la propia SGAE ya que lo que no se puede hacer es cambiar la naturaleza intrínseca de los buscadores que ordenan la información en función de las valoraciones de los usuarios de Internet. Las primeras posiciones de los resultados no dependen de la opinión de los responsables del buscador (de ser así tendríamos en dictadura electrónica), si no de las de aquellos que quieren participar en su clasificación. Puesto que actualmente, las posturas de los usuarios de la red son bastante contrarias a la política que lleva a cabo la SGAE, es lógico que éstas se reflejen a la postre en los buscadores de información.

Actualmente, si realizas una búsqueda en un Google con el término "ladrones" podrás ver en la parte inferior de la página de resultados el texto:



Pese al ruido montado la SGAE sigue apareciendo al buscarse la palabra 'Ladrones'.



Desde el blog de Metroseo se explica la técnica del Wiki-dnapping.



"En respuesta a un requisito legal enviado a Google, hemos eliminado 2 resultado(s) de esta página. Si lo desea, puede leer más información sobre este requisito en ChillingEffects.org".

El enlace nos lleva a la página "chillingeffects.org", el sitio web que sirve de depositario de las denuncias relacionadas, sobre todo, con derechos de autor (ver ejemplos de DMCA o periódicos belgas) que expone escuetamente en un texto en inglés "Spanish defamation complaint to Google. The notice is not available" (demanda española por difamación a Google. El aviso no está disponible).

En cualquier caso la SGAE también ha tomado sus propias medidas tecnológicas para evitar posteriores problemas de esta índole y no depender únicamente de la acción de los responsables del buscador. Para ello, y aunque sea tras varios años, la Sociedad ha insertado un fichero robots.txt (http://www.sgae.es/robots.txt) en su página. Los contenidos de este fichero son:

```
User-agent: *    # aplicable a todos
Disallow:        # permite la indexacion de
                   todas las paginas
```

Este tiene como finalidad impedir a todos los robots (las aplicaciones virtuales que se encargan de cartografiar la red y sus contenidos) de todos los buscadores rastrear cualquier documento situado bajo el dominio "www.sgae.es", excluyendo de los resultados del buscador web de Google las páginas que aparecen en primeras posiciones para la consulta de "ladrones".

Eso no quita para que, en un principio y a pesar de que los responsables del sitio web de la SGAE eran conscientes de que utilizar un fichero "robots.txt" sería muy apropiado para este tipo de ataques de Google Bombings, prefirieron acusar a Google de las opiniones de los usuarios, enviando además denuncias a algunos bloggers.

Este caso sigue haciendo correr ríos de tinta y con toda seguridad volveremos a oír hablar de él ya que entran en liza varios factores muy interesantes como lo son el derecho a libertad de expresión, el poder de la masa social o la evolución de las nuevas tecnologías. De momento, si buscamos el término "ladrones" en Google, sigue apareciendo en segunda posición la página web de la SGAE, y existe una iniciativa que ha exportado el modelo virtual de Google Bombing hacia la Sociedad General de Autores al mundo real, escribiendo en los billetes de euro la consabida consigna para que esta se transfiera de mano en mano.

La Wikipedia

Indudablemente, la enciclopedia libre conocida como Wikipedia es por derecho propio uno de los sitios web con más éxito de la red de redes, un hecho que genera a su vez un círculo vicioso que la retroalimenta debido al excelente posicionamiento que sus páginas tienen dentro de los resultados del buscador web de Google. Muchos creadores de contenidos web (sobre todo, bloggers) no dudan en enlazar a los artículos de la Wikipedia cada vez que quieren que los lectores sepan algo más sobre determinado tema, lo que provoca que la relevancia que le otorga Google a dicho artículo aumente (unido a los enlaces internos y a la confianza que el dominio "wikipedia.org" ha conseguido para Google) en una espiral de popularidad que le ha permitido situarse en los primeros puestos.



Google muestra ante determinadas búsquedas que ha eliminado algunos resultados.

Sin embargo, esta situación de populismo virtual no es del agrado de ciertos webmasters que ven cómo los artículos de la Wikipedia se posicionan por delante de unas páginas web con temáticas muy jugosas económicamente que han creado y diseñado con mucho esmero, trabajando en su posicionamiento y pidiendo enlaces para poder situarlas en los primeros puestos.

Hace algunos años, muchos especialistas en posicionamiento web de los llamados "black hat" (que hacen uso de técnicas penalizadas) llegaron a editar incluso los propios artículos de la Wikipedia para así obtener enlaces directos hacia sus páginas a posicionar. Esto provocó que los responsables del proyecto tomaran la medida de comenzar a insertar el parámetro "rel=nofollow" en estos enlaces, una técnica que impide que Google lo siga y, por lo tanto, que lo consideren como un "voto" en su algoritmo de popularidad.

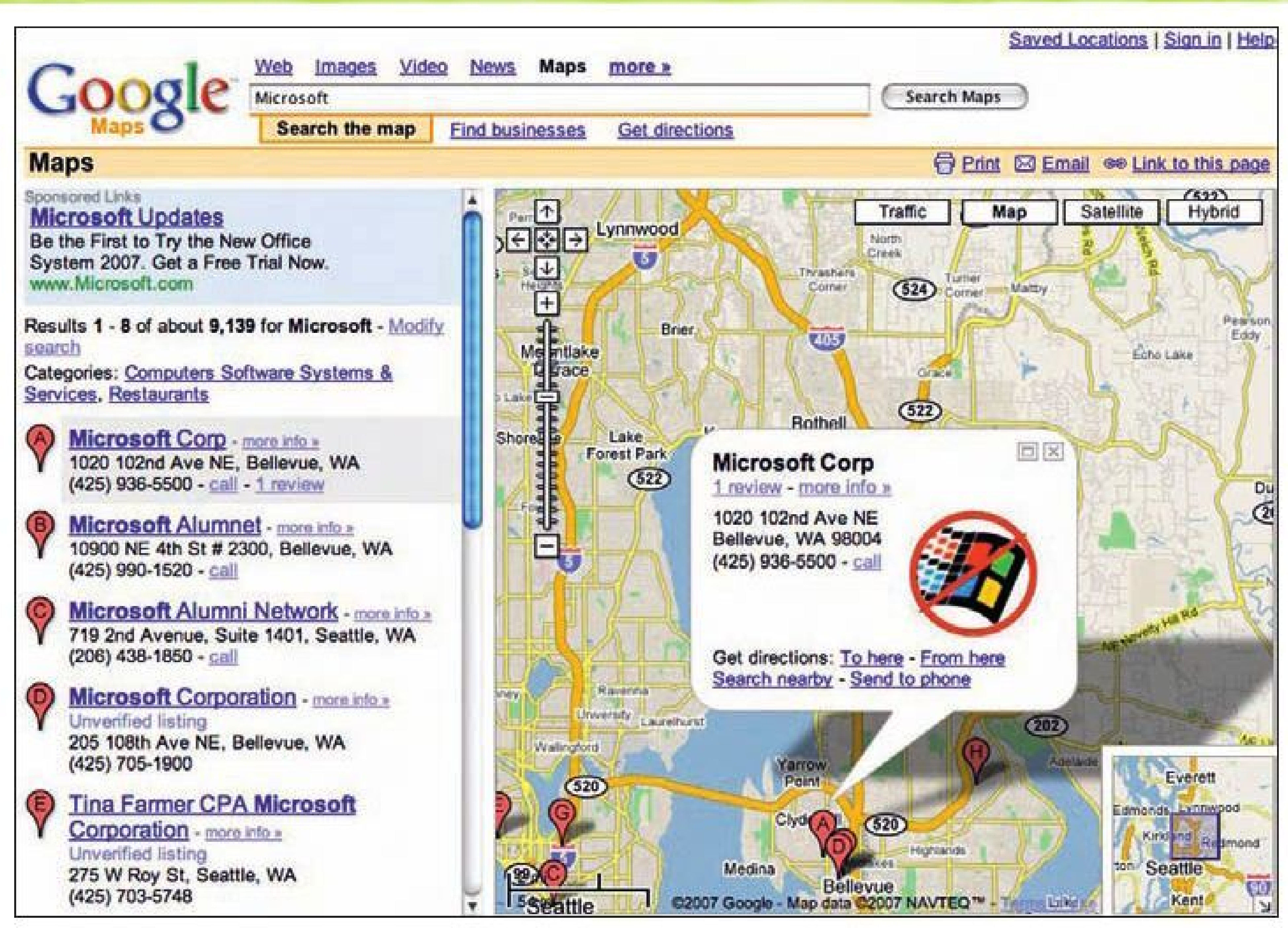
Para remediar este "contratiempo", desde el blog Metroseo (metroseo.com) - dedicado a estrategias SEO (search engine optimization u optimización de motor de búsqueda) para mejorar el posicionamiento web - se detalla cómo hacer para editar los artículos de la Wikipedia para lograr que estas páginas pierdan algo de posicionamiento y que las del interesado aparezcan por encima de esta. La técnica conocida como "Wiki-dnapping" consiste en crear varios usuarios que gocen de credibilidad (añadiendo información valiosa, eliminando spam, colaborando con el proyecto), y dedicarse con frecuencia aunque disimuladamente a eliminar enlaces internos dentro de la propia Wikipedia para así empeorar su posicionamiento final. Según aseguran los responsables de este blog, esto no empeora la calidad de la Wikipedia, aunque sí inflige un cierto castigo a las páginas de un proyecto que insertan el famoso "rel=nofollow" a los enlaces externos que incluyen los usuarios, pero no hace lo mismo con los links hacia empresas privadas afiliadas a los dirigentes de la Wikipedia.

Del mismo modo existe la vertiente contraria, la de los webmasters cuyo objetivo es que los artículos de la Wikipedia estén por encima... pero de las páginas de su competencia, con el objeto de perjudicarlas para que estas pierdan tráfico de red y, lo que ello conlleva, ingresos. Por ejemplo en la página de Seomoz (seomoz.org) se anima a elegir el artículo de la Wikipedia que más se ajuste a las palabras de la búsqueda que se desee manipular (si no existe, se crea), a obtener un par de enlaces hacia él y esperar. Dado que el posicionamiento de las páginas de la Wikipedia es tan efectivo, no necesitamos más.

Dos dinámicas que vuelven a reabrir el debate sobre si la relevancia que Google otorga a los artículos de la Wikipedia es desmesurada o no, y si los usuarios realmente quieren encontrarse con su información cada vez que realizan una búsqueda determinada. Para muestra un botón: hace unos años se llevó a cabo un Google Bombing para intentar que una página de la Wikipedia apareciese en las primeras posiciones al buscar "online poker" en Google, una información que sin duda no es la que muchas personas quieren encontrar, ¿verdad?

¿Punto final... o punto y aparte?

Tras varios casos sonados, entre los que destaca tristemente el de la SGAE, en enero de 2007



Los mapas de Google ya han sufrido algún caso de una curiosa variante de Google Bombing.

Google anunciaba que pondría fin a los ataques de Google Bombings mediante la inclusión de una serie de filtros manuales creados explícitamente para evitar que ciertas páginas web apareciesen en la primera posición de los resultados del buscador web al consultar ciertos términos. Anunciaba que se había comenzando a minimizar el efecto de muchos de los Google Bombings dentro de las páginas de resultados del buscador web y, para ello se había "mejorado el análisis de la estructura de enlaces de la red".

Sin embargo, al tratarse de filtros diseñados a mano, se trata de un método poco adaptable, haciendo que otras búsquedas no tenidas en cuenta inicialmente comiencen a devolver también la "página víctima" debido a los numerosos enlaces recibidos por ésta, y de los que Google parece no ser capaz de sortear, por el momento.

Así por ejemplo, si buscamos ahora "miserable failure" en Google, efectivamente no aparece la página web del presidente George W. Bush. Por el contrario, si sólo buscamos "failure", sí que pudo encontrarse durante cierto tiempo la página en cuestión en la primera posición.

El motivo es bien sencillo: Google sigue teniendo en cuenta los miles de enlaces con el texto "miserable failure" hacia "www.whitehouse.gov/president/", al igual que lo hace con la palabra "ladrones" y el enlace a la página www.sgae.es. Si encima, se da la circunstancia de que el término en cuestión aparece dentro del texto de los contenidos de dicha página, el resultado se ve ratificado, asignándosele a esa consulta un nivel de relevancia adicional que no hace si no ratificar su posicionamiento.

En su momento, Google anunció que, aunque no le gustara este tipo de prácticas, tampoco tenía intención de eliminar los resultados erróneos manualmente excusándose en que el Google Bombing era más bien una forma de entretenimiento para algunos ciertamente inocuo y que nunca llegaría a alterar la calidad intrínseca del buscador. Inclusive, desde un principio ha defendido que esta técnica tiene éxito debido al propio funcionamiento del buscador, es decir, acepta la vulnerabilidad de su sistema con respecto a este tipo de ataques, aunque dejando muy claro que estos son posibles debido únicamente a la propia eficacia de su algoritmo de búsquedas. Esto es, si existe el Google Bombing es porque su sistema de posicionamiento es el mejor.

Algunos Google Bombings son permitidos por el buscador, quizá porque se considere que no son tan políticamente incorrectos, o quizá porque se apoyan en el marketing viral para seguir aupando a Google como el rey de los buscadores. Sin embargo, desde que oficialmente se pusiese fin a los ataques de Google Bombing por parte del buscador, son muchos los que añoran la existencia de estas iniciativas que ponían a prueba la repercusión de una determinada protesta a través de Internet.

De ahí que algunos usuarios sigan buscando la manera de "saltarse" esta nueva barrera impuesta por los responsables de Google, o de crear un sistema alternativo que permita resultados similares. Así hemos podido encontrar un nuevo tipo de GoogleBomb creado con los mapas de Google modificando la insignia de las jefaturas de Microsoft a una imagen de contra de Windows. Aunque la imagen ya no está mucha gente capturó la pantalla. El procedimiento fue muy sencillo: "Sara B", un usuario de Internet, subió una imagen en contra de Windows, la imagen fue puesta en la primera posición respecto al motor de búsqueda de Google y fue exhibida en los mapas de Google para las jefaturas de Microsoft.

Otro método alternativo es el que aparece en este blog: "http://eloi.programacionweb.net/blog/post.php?id=139"

Desde aquí se lanza la hipótesis de que el nuevo algoritmo hace que las páginas web que no contengan la palabra enlazada (y posteriormente buscada) sean penalizadas a la hora de calcular su "LocalRank" y, por lo tanto, a la hora de aparecer en las páginas de resultados del buscador.

El "LocalRank"' es una teoría que trata de explicar una segunda clasificación de las páginas web (tras la que calcula el "PageRank") dentro de los resultados, y que estaría potenciada por diferentes factores, como por ejemplo el hecho de conseguir enlaces desde sitios web de la misma temática.

Para sortear esta penalización, se asegura, un posible truco podría ser incluir en la URL de la "víctima" la palabra clave que queremos posicionar con lo que se conoce como términos falsos (por ejemplo: "http://dominio.com/?palabra_clave"). Sin embargo nos encontraríamos con dos inconvenientes. El primero, y el más importante, es que Google aseguraba hace unos días que modificaba "a mano" los resultados de las búsquedas que se proponen en los Google Bombings, por lo que lo podrían evitar una vez la iniciativa consiguiese una cierta repercusión. Y el segundo inconveniente es que la URL propuesta, al contrario que las que no contienen la palabra a posicionar, puede ser "bloqueada" por el responsable del sitio web mediante el uso de robots (como el que ya utiliza a la SGAE). Este método que utiliza lo que se denomina falsos parámetros muestra en muchos sitios web del enlace: http://www.sgae.es/?ladrones.

Y es que hecha la ley, hecha la trampa. 

Enlaces

- Wikipedia: http://es.wikipedia.org/wiki/Google_bomb
- Noticias Google: http://google.dirson.com/google-bombing.php
- Blog Eloi de San Martin: http://eloi.programacionweb.net/blog/post.php?id=139
- Demanda SGAE: http://www.merodeando.com/2007/02/20-a-la-sgae-no-le-gusta-merodeando
- SGAE: http://www.sgae.es
- PageRank: http://es.wikipedia.org/wiki/PageRank
- http://www.20minutos.es/noticia/205066/0/sgae/ladrones/socio/



RSS con Java (III)

ADOLFO ALADRO GARCÍA

Las aplicaciones Web de los servicios y portales que ofrecen RSS pueden verse sometidas a un gran estrés debido a la gran cantidad de documentos que tienen que servir simultáneamente. Por ello es indispensable implementar políticas de optimización que hagan que las aplicaciones puedan soportar esta carga con unos niveles aceptables de rendimiento.

Los mecanismos de optimización de las aplicaciones *Web* responsables de servir documentos *RSS* son muy diversos y dependientes de las características propias del servicio. Por ejemplo, no es lo mismo optimizar un portal de noticias que sirve documentos *RSS* que un servicio de *blogs*. Ahora bien, existen elementos comunes en todos los desarrollos que están más allá de la especificidad de cada servicio, y cuya optimización es vital si se quiere garantizar un rendimiento razonable. Fundamentalmente se pueden optimizar dos tareas: la de creación de los documentos *RSS* y la de servirlos. Para lo primero se va a estudiar de qué manera puede mejorarse la clase *RssDocument* con el fin de evitar la creación constante de objetos que son susceptibles de poder ser reutilizados. Para lo segundo se va a analizar la creación de sistemas de caché que eviten tener que crear el mismo documento *RSS* repetidas veces en un corto espacio de tiempo.

Fig. 1. Fichero web.xml de la aplicación Web de ejemplo.

LA CLASE DocumentBuilderPool

Una de las primeras optimizaciones que pueden realizarse cuando se trabaja con el *DOM* de *XML* consiste en evitar la constante creación de instancias de las clases *DocumentBuilderFactory* y *DocumentBuilder*, ambas pertenecientes al paquete *javax.xml.parsers* del *API* estándar de *Java*. La clase *DocumentBuilderPool* recoge este requerimiento ya que crea un *pool* de instancias de objetos de tipo *DocumentBuilder* que se reutilizan. Esta clase cuenta con dos métodos principales:

```
public final synchronized DocumentBuilder
checkOut()
```

```
public final synchronized void
checkIn(DocumentBuilder docBuilder)
```

Cuando se necesita un objeto de tipo *DocumentBuilder* se llama al método sincronizado *checkOut*. Posteriormente, cuando se ha terminado de utilizar dicho objeto, se devuelve al *pool* empleando el método, también sincronizado, *checkIn*. Así los objetos *DocumentBuilder* son reutilizados entre distintas llamadas.
Para implementar la clase *DocumentBuilderPool* se usa la clase *LinkedHashMap*, perteneciente al paquete *java.util* del *API* estándar de *Java*, ya que cuenta con un método muy interesante denominado *removeEldestEntry*. Éste se encarga de eliminar aquellas las entradas más antiguas bajo determinadas circunstancias. La implementación es realmente sencilla: (ver Listado 1)
La clase *LRULinkedHashMap* extiende a la clase estándar *LinkedHashMap* con el fin de implementar el método *removeEldestEntry*. El constructor recibe un parámetro que se corresponde con el número máximo de entradas que la estructura de datos va a contener. Cada vez que se usa la instancia de *LRULinkedHashMap* (en realidad la instancia de *LinkedHashMap*) de forma transparente al programador se registra dicha actividad y se termina llamando al método *removeEldestEntry*. Si este método devuelve el valor *true* significa que se puede eliminar de la estructura de datos la entrada que menos se haya usado recientemente. De ahí el prefijo *LRU*, que en inglés significa *Last*

Material adicional

El material complementario puede ser descargado desde nuestra web www.revistasprofesionales.com



LISTADO 1

```
private final static class LRULinkedHashMap<T,V>
extends LinkedHashMap<T,V>
{
  private static final long serialVersionUID = 1L;
  private final int iMaxSize;

  public LRULinkedHashMap(int iMaxSize) {
    super(iMaxSize * 4 / 3 + 1, 0.75f, true);
    this.iMaxSize = iMaxSize;
  }

  public final boolean removeEldestEntry(Map.Entry eldest) {
    return size() > iMaxSize;
  }
}
```

LISTADO 2

```
private DocumentBuilderPool(int iSize)
throws Exception {
  mapUsed = new LRULinkedHashMap<DocumentBuilder,Boolean>(iSize);
  mapUnUsed = new LRULinkedHashMap<DocumentBuilder,Boolean>(iSize);

  docBuilderFactory = DocumentBuilderFactory.newInstance();
  docBuilderFactory.setValidating(false);

  for(int i=0; i<iSize; i++) {
    mapUnUsed.put(docBuilderFactory.newDocumentBuilder(), Boolean.TRUE);
  }
}
```

LISTADO 3

```
for (final DocumentBuilder docBuilder: mapUnUsed.keySet()) {
  mapUnUsed.remove(docBuilder);
  mapUsed.put(docBuilder, Boolean.TRUE);
  return docBuilder;
}

final DocumentBuilder docBuilder = docBuilderFactory.newDocumentBuilder();
mapUsed.put(docBuilder, Boolean.TRUE);
return docBuilder;
```

Recentley Used. De esta forma se garantiza que independientemente del uso, el número de objetos que almacena la instancia de *LRULinkedHashMap* se mantiene constante en el tiempo.

Continuando con la implementación de *DocumentBuilderPool*, se puede observar que la clase cuenta con tres atributos de tipo *final*:

```
private final LRULinkedHashMap<
DocumentBuilder,Boolean> mapUsed;
```

```
private final LRULinkedHashMap<
DocumentBuilder,Boolean> mapUnUsed;
```

```
private final DocumentBuilderFactory
docBuilderFactory;
```

La primera estructura de datos contendrá aquellas instancias de la clase *Document Builder* que se estén usando en cada momento. La segunda contendrá aquellas otras que no se estén usando, es decir, las que estén libres. Finalmente el atributo *docBuilderDactory* se empleará para crear nuevas instancias de la clase *Document Builder*. Todos estos atributos se inicializan, como es lógico, en el constructor de la clase: (ver Listado 2)

Así las estructuras de datos se inicializan con un parámetro que se corresponde con el número máximo de elementos que tendrán a lo largo de su existencia, se crea una instancia de *DocumentBuilderFactory* y finalmente se crean los objetos *DocumentBuilder* añadiéndolos a la estructura de datos que guarda aquellos que están libres.

Los métodos *checkOut* y *checkIn* son realmente simples. El primero explora la estructura de objetos *DocumentBuilder* que no están en uso. Si encuentra alguno lo devuelve. En caso contrario crea un nuevo objeto *DocumentBuilder*, lo añade a la cola de los que están en uso y lo devuelve: (ver Listado 3)

El método *checkIn* simplemente devuelve el objeto *DocumentBuilder* al conjunto:

```
docBuilder.reset();
```

```
mapUsed.remove(docBuilder);
```

```
mapUnUsed.put(docBuilder,
Boolean.TRUE);
```

Obsérvese que tanto *checkout* como *checkIn* están sincronizados. Esto es esencial ya que los *servlets* se ejecutan concurrentemente en el servidor de aplicaciones. Con esta clase cuando una llamada termina de ejecutarse el correspondiente objeto *DocumentBuilder* no se descarta, con las consiguientes tareas internas que esto conlleva y que afectan a la memoria de la Máquina Virtual *Java*. Al contrario, el objeto vuelve al *pool* de tal forma que en la siguiente llamada podrá ser reutilizado.

Con el fin de utilizar la clase *Document BuilderPool* los cambios que hay que hacer en la clase *RssDocument* creada en la entrega anterior, son mínimos. En vez de crear constantemente objetos *Document Builder* simplemente se cogen y devuelven del *pool*:

```
DocumentBuilder docBuilder =
DocumentBuilderPool.getInstance().
checkOut();
```

```
doc = docBuilder.newDocument();
```

```
DocumentBuilderPool.getInstance().che
ckIn(docBuilder);
```

El último aspecto que debe estudiarse es el que respecta a la propia existencia del *pool*. Lo lógico es pensar que la aplicación siempre debe manejar la misma instancia de la

LISTADO 4

```
private final static DocumentBuilderPool documentBuilderPool = newInstance();

private final static DocumentBuilderPool newInstance(){
  try {
    return new DocumentBuilderPool(8);
  } catch (Exception e) {
    return null;
  }
}
```





Fig.2. Resultado que muestra el navegador Firefox al cargar el documento RSS devuelto por el servlet SRss Server3.

clase *DocumentBuilderPool*. Esto, en lenguaje de programación orientado a objetos, es lo que se denomina *singleton*, y la razón por la que el constructor de *Document BuilderPool* es privado. La única forma de obtener una instancia es utilizando el método estático *getInstance*. Una variable estática y privada contiene una instancia de la propia clase, instancia que se ha creado justo cuando la clase va a ser utilizada: (ver Listado 4)

El método *getInstance* simplemente permite acceder a la variable estática y privada, de forma que siempre se está usando una única instancia de *DocumentBuilderPool*:

```
public final static
DocumentBuilderPool getInstance() {
  return documentBuilderPool;
}
```

LA CLASE RssDocumentPool

El mecanismo anterior puede extenderse para crear un *pool* o una caché de objetos *RssDocumentPool*. El objetivo es fácil de entender. Lo más común es que en un corto espacio de tiempo varios usuarios soliciten el mismo documento *RSS*. No tiene sentido crear el mismo documento una y otra vez para cada una de esas peticiones, y esto se aplica especialmente en aquellos casos en los que el documento se crea accediendo a una base de datos. Es más, no tiene ni siquiera sentido guardar en una caché el objeto *RssDocument* tal cual, ya que también se penalizaría innecesariamente el rendimiento de la aplicación al tener que estar utilizando el método *write* para obtener la cadena de texto correspondiente al objeto *RssDocument*. Es decir, lo más interesante es guardar en la caché directamente esa cadena de texto, y ése es el propósito de la clase *RssDocumentPool*.

Aunque la clase *RssDocumentPool* utiliza un mecanismo similar al de la clase *Document BuilderPool*, hay algunas diferencias. Para empezar *RssDocumentPool* sólo cuenta con un atributo:

```
private final LRULinkedHashMap<String,
String> mapLRUCache;
```

Lo que se almacena en *mapLRUCache* son pares de cadena de texto, de forma que el primero, la clave (*key*), es el identificador del documento *RSS*, y el segundo, el valor (*value*), la cadena de texto correspondiente al *XML*.

Los dos métodos principales de *Rss DocumentPool* se denominan *get* y *put*:

```
public final synchronized String
 get(String sRssDocId)
public final synchronized void
put(String sRssDocId, RssDocument rssDoc)
```

El primero recibe el identificador del documento *RSS* y devuelve la cadena de texto correspondiente al *XML* del documento *RSS*. El segundo recibe el identificador y el objeto *RssDocument* y guarda la información. La implementación del método *get* no puede ser más simple. Si el documento *RSS* está en la caché, devuelve la cadena correspondiente al *XML* del mismo. En caso contrario devuelve *null*:

```
return mapLRUCache.get(sRssDocId);
```

El método *put* tiene más código ya que es necesario transformar el objeto *Rss Document* en la correspondiente cadena de texto antes de pasar a almacenar la información en la caché. Para ello se emplean las clases *BufferedWriter* y *StringWriter*. La primera optimiza los accesos de escritura ya que utiliza un *buffer* interno. La segunda sirve para obtener la cadena de texto escrita cuando la operación ha terminado: (ver Listado 5)

El proceso de obtener la cadena de texto correspondiente al *XML* del documento *RSS* se hace empleando el típico bloque *try...catch...finally* que garantiza que al final los canales de escritura se cierran y los recursos se liberan, con independencia de lo ocurrido. Si *sRssDocAsStr* es distinto de *null*, entonces se guarda en la caché usando además el identificador, *sRssDocId*.

El *servlet SRssServer4* ilustra el uso de la clase *RssDocumentPool*. Primeramente el método *service* se ha modificado para que muestre por la consola el tiempo que tarda así como si se ha utilizado la caché: (ver Listado 6)

Primeramente se obtiene una instancia (en realidad la única que existe ya que la caché es objeto *singleton*, igual que sucedía con la clase *DocumentBuilderPool*) de la clase *RssDocumentPool*:

```
RssDocumentPool rssDocPool =
RssDocumentPool.getInstance();
```

LISTADO 5

```
String sRssDocAsStr = null;
BufferedWriter bw = null;
StringWriter sw = null;
try {
  sw = new StringWriter();
  bw = new BufferedWriter(sw);
  rssDoc.write(bw);
  bw.flush();
  sw.flush();
  sRssDocAsStr = sw.toString();
} catch (Exception e) {
  e.printStackTrace();
} finally {
  try {if (sw!=null) sw.close(); sw=null;} catch (Exception e) {}
  try {if (bw!=null) bw.close(); bw=null;} catch (Exception e) {}
}
if (sRssDocAsStr!=null) {
  mapLRUCache.put(sRssDocId, sRssDocAsStr);
}
```

LISTADO 6

```
public void service(HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException {
  final long l = System.currentTimeMillis();
  boolean bCached = false;
  ...
  System.err.println("SRssServer4 (cached=" + bCached + "): " +
(System.currentTimeMillis() - l) + "ms.");
}
```



Seguidamente se intenta obtener de la caché la cadena de texto correspondiente al *XML* del documento *RSS* con el método *get*:

```
String sRssDocAsStr =
rssDocPool.get("modigliani");
```

En este ejemplo la cadena *modigliani* es el identificador del documento *RSS*.

Si no se tiene éxito, es decir, si el documento no está en la caché, entonces se crea tal y como se ha hecho en los ejemplos anteriores, y se guarda en la caché: (ver Listado 7)

Si la caché guarda el documento, no hay que hacer nada. Al final del bloque anterior simplemente queda escribir en el canal de escritura del *servlet* la cadena de texto guardada en la variable *sRssDocAsStr*:

```
bw.write(sRssDocAsStr);
```

Si se hacen repetidas llamadas a la *URL* que sirve el documento *RSS* utilizando el *servlet SRssServer4* y se sigue el *log* del servidor de aplicaciones se puede observar:

```
SRssServer4 (cached=false): 47ms.
SRssServer4 (cached=true): 0ms.
SRssServer4 (cached=true): 0ms.
SRssServer4 (cached=true): 0ms.
```

Cuando el documento no está en la caché el tiempo que tarda en servirse en mayor ya que hay que crear el documento *RSS* con la clase *RssDocument*, obteniendo después la cadena de texto correspondiente al *XML*. A la luz de esta pequeña prueba cualquier podría decir que la diferencia entre los tiempos en uno y otro caso es despreciable. Ahora bien, hay que tener en cuenta que en un entorno real el número de documentos *RSS* diferentes puede ser grande, que el tamaño de los mismos será igualmente considerable y sobretodo, que el número de accesos concurrentes puede ser altísimo. En esas condiciones este tipo de caché puede marcar la diferencia entre una aplicación lenta y sobrecargada o una que da servicio de una forma razonable. Además el tipo de caché empleado ofrece una ventaja considerable. Se trata de una caché *LRU* (*Last Recently Used*), es decir, que sólo se almacena un número fijo de documentos, concretamente aquellos que se usan con mayor frecuencia. Esto significa que la memoria necesaria para la caché está controlada, y no sólo controlada sino optimizada, en el sentido de que con el tiempo en la caché siempre están aquellos documentos

LISTADO 7

```
if (sRssDocAsStr==null) {
  ...
  rssDocPool.put("modigliani", rssDoc);
  sRssDocAsStr = rssDocPool.get("modigliani");
} else {
  bCached = true;
}
```

LISTADO 8

```
RssDocumentCacheable rssDocCacheable = mapLRUCache.get(sRssDocId);
if (rssDocCacheable!=null &&
rssDocCacheable.hasExpired(System.currentTimeMillis())) {
  mapLRUCache.remove(sRssDocId);
  rssDocCacheable = null;
}
return rssDocCacheable;
```



Fig.3. Resultado que muestra el navegador Internet Explorer al cargar el documento RSS devuelto por el servlet SRssServer3.

RSS que por su popularidad son más usados. Si se piensa por ejemplo en un servicio de *blogs*, con cientos y cientos de *blogs*, la aplicación tiene que servir muchos documentos *RSS*. No sería viable tener todos esos documentos en la caché ya que la memoria necesaria sería demasiado grande. Ahora bien, de entre todos esos *blogs* seguramente habrá unos pocos que sean realmente muy populares, y el resto tendrán una audiencia menor. Basta con optimizar el acceso a los documentos *RSS* de los más populares para optimizar la aplicación.

LA CLASE RssDocumentCache

Casi siempre que se construye una caché hay que tener en cuenta que los objetos almacenados pueden quedar desfasados, es decir, que existe un tiempo más allá del cual la validez de los mismos expira. La clase *RssDocumentPool* expuesta en el apartado anterior no contempla esta posibilidad ya que no comprueba los objetos que almacena. Con este mecanismo los objetos son válidos siempre: si están en la caché, ésta los devuelve, y en caso contrario se crea el objeto de nuevo. Tratándose de documentos *RSS* esto no tiene mucho sentido ya que su actualización, al menos diaria, suele ser frecuente. La clase *RssDocumentCache* ilustra una forma simple sobre cómo introducir el concepto de expiración.

La primera diferencia con respecto a *RssDocumentPool* es que los objetos que almacena *RssDocumentCache* implementan la interfaz *RssDocumentCacheable*.

```
public interface RssDocumentCacheable {
  public String getId();
  public boolean hasExpired(long lTime);
  public String toString();
}
```

El método *getId* de la interfaz devuelve el identificador del objeto. El método *hasExpired* devuelve *true* o *false* que indica si el objeto ha expirado. El parámetro que recibe, *lTime*, es un valor numérico que indica una fecha y hora, en forma de número de milisegundos. Finalmente el método *toString* devuelve la cadena de texto del *XML* correspondiente al documento *RSS*.

Ahora, el metodo *get* de la clase *RssDocumentCache* no se limita a devolver el valor almacenado, si es que está, sino que además verifica que éste no haya caducado: (ver Listado 8)

Documentación de la clase LinkedHashMap del API estándar de Java.



LISTADO 6

```
rssDocCacheable = new RssDocumentCacheable() {
  private String sRssDocAsStr = null;
  private long    l = System.currentTimeMillis();

  public final String getId() {
    return "modigliani";
  }
  public final boolean hasExpired(long lTime) {
    return (lTime - l) > 10000;
  }

  public final String toString() {
    if (sRssDocAsStr==null) {
      ...
      this.sRssDocAsStr = sRssDocAsStr;
    }
    return sRssDocAsStr;
  }
};
```

Fig.5. Código que garantiza que la clase Rss DocumentPool es un singleton, es decir, que sólo puede existir una instancia de la misma en la aplicación.

Si el objeto está almacenado en la caché y ha expirado, es como si no estuviera. Por lo tanto se borra de la caché y el valor devuelto es *null*. Existen muy diversas formas de implementar este tipo de mecanismos. Otra podría haber sido devolver el objeto en cualquier caso y que fuera la clase que utiliza la caché la responsable de verificar la validez del objeto de la caché. En realidad las diferencias entre unos modelos y otros no representan gran diferencia, y la idea es la misma. El objeto *RssDocument* podría modificarse para que implementara la interfaz *RssDocumentCacheable*. También es posible crear dinámicamente los objetos que implementan esta interfaz. El *servlet SRssServer5* muestra cómo hacerlo, partiendo de la variable *rssDoc* que es de tipo *final*, y que se ha creado como en los ejemplos anteriores: (ver Listado 9)

El método *getId* devuelve la cadena *modigliani*, que es el identificador del documento *RSS*. Lógicamente en un entorno real ese identificador se obtendría de otra forma, pero para el ejemplo se puede simplificar. El método *hasExpired* es el más interesante. Toma el parámetro que recibe y comprueba cuánto tiempo ha transcurrido. Si han pasado más de 10 segundo (10000 milisegundos) se considera que el objeto ha caducado. Finalmente el método *toString* construye la cadena de texto correspondiente al *XML* del documento *RSS* utilizando el mismo mecanismo empleado en la clase *Rss DocumentPool*. Ahora bien, en vez de construirla en cada llamada, la construye únicamente la primera vez, y el resto de las veces simplemente devuelve el valor de su atributo *sRssDocAsStr*. Es decir, esta cadena también está cacheada dentro del propio objeto que implementa la interfaz *RssDocument Cacheable*.

Con estas modificaciones se puede observar el resultado de nuevo echando un vistazo al *log* del servidor de aplicaciones:

```
SRssServer5 (cached=false): 31ms.
SRssServer5 (cached=true): 0ms.
SRssServer5 (cached=true): 0ms.
SRssServer5 (cached=true): 0ms.
SRssServer5 (cached=true): 0ms.
SRssServer5 (cached=true): 0ms.
SRssServer5 (cached=false): 32ms.
SRssServer5 (cached=true): 0ms.
SRssServer5 (cached=true): 0ms.
SRssServer5 (cached=true): 0ms.
SRssServer5 (cached=true): 0ms.
...
```

La primera vez la caché está vacía, por lo que el objeto *RssDocument* se crea. Asimismo es necesario crear la cadena de texto correspondiente al *XML*. Por lo tanto el tiempo que tarda la llamada es considerable (De nuevo hay que insistir en que los tiempos que se ilustran son relativos en el sentido de que en cualquier caso son pequeños porque no se trata de un entorno real). Durante un tiempo las subsiguientes llamadas devuelven lo almacenado en la caché. Cuando transcurren

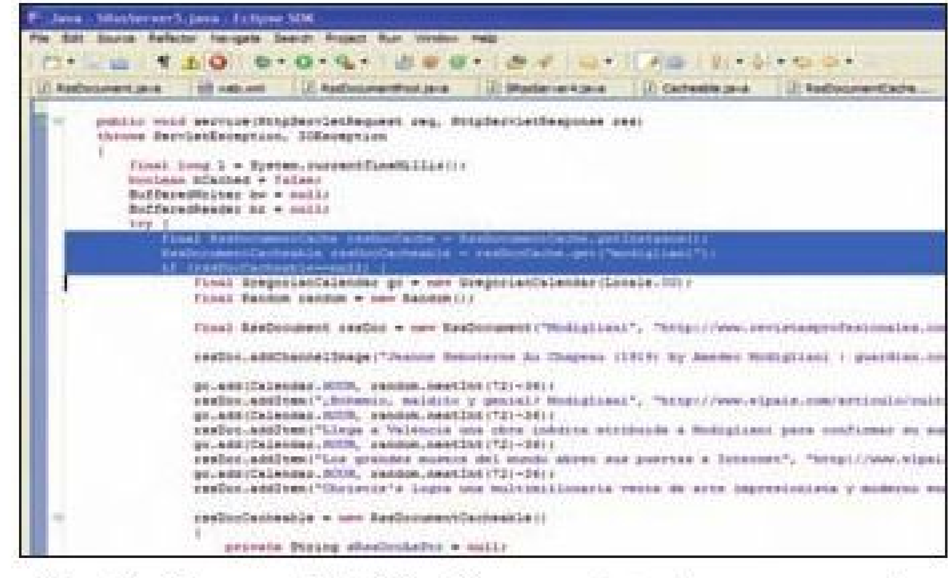


Fig.6. El servlet SRssServer5 primeramente intenta obtener el objeto de la caché.

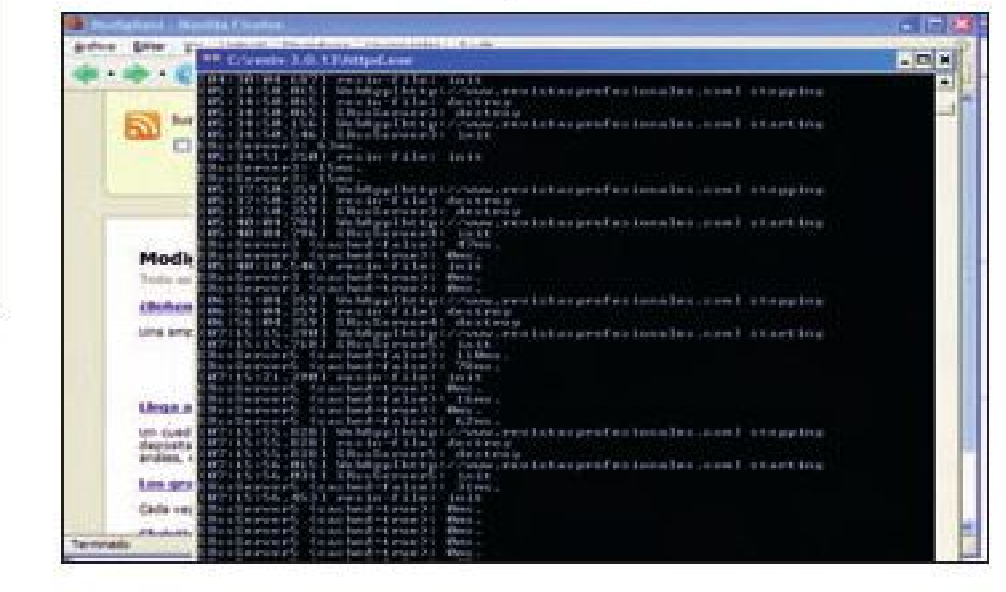


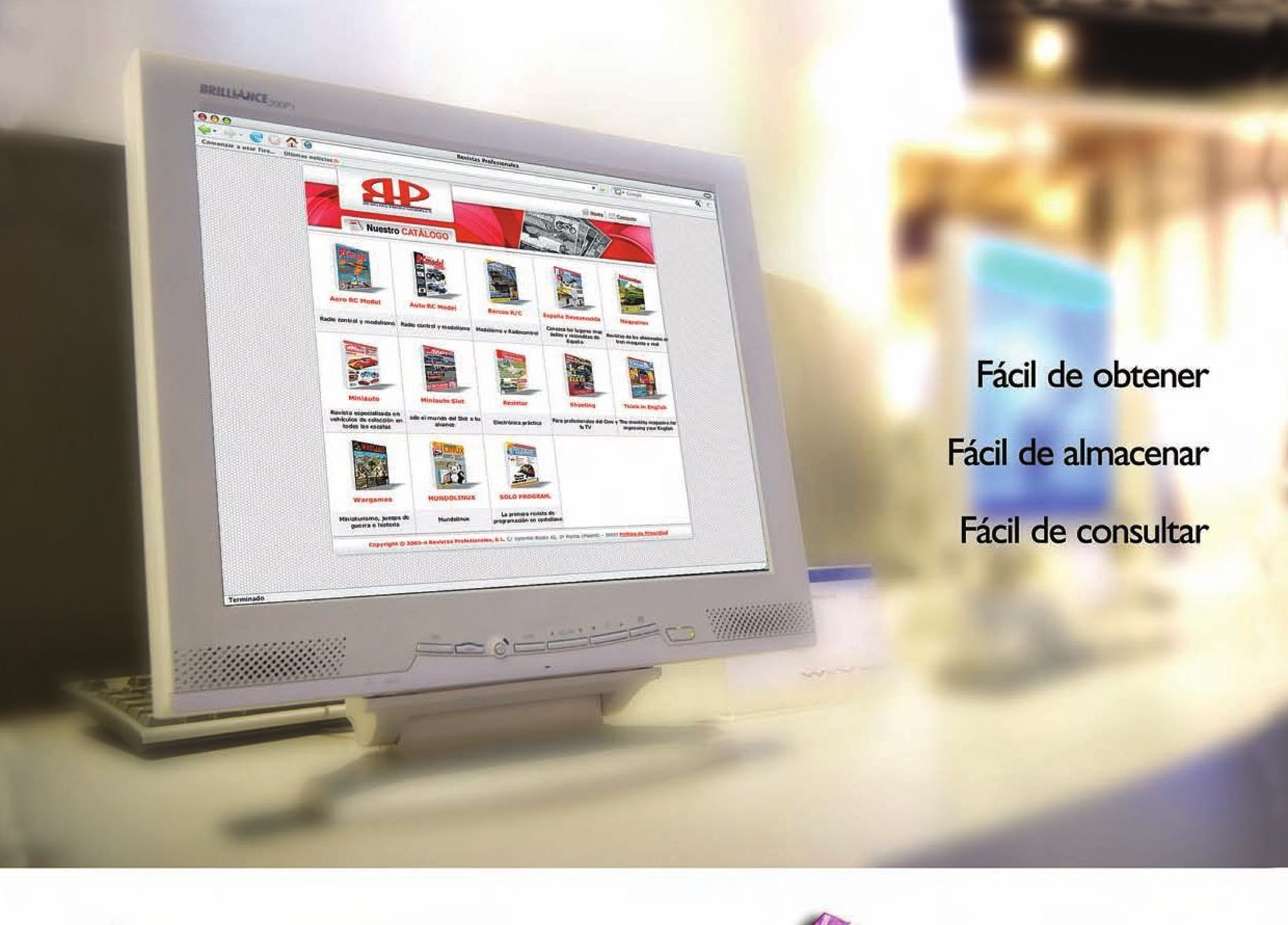
Fig. 7. Salida de la consola del servidor de aplicaciones Web.

10 segundos, la caché detecta que el objeto ha caducado y por lo tanto da lugar a que se cree de nuevo, volviéndolo a almacenar, eso sí, ya actualizado.

El mecanismo empleado para la validación del documento *RSS* es abstracto. Si el origen de datos fuese por ejemplo unos ficheros, el método *hasExpired* podría utilizar la fecha de creación o actualización de los ficheros. Si se emplea una base de datos, podría consultarse la última fecha de actualización, de forma que los accesos a la misma quedarían sustancialmente reducidos.

CONCLUSIÓN

En este punto ya se ha desarrollado una idea clara de cómo es posible optimizar la creación de los documentos *RSS* así como la tarea de servirlos. Pero todavía se puede ir más allá, como se comprobará en la siguiente entrega. Las cabeceras *HTTP* estándar desempeñan un papel importante a la hora de servir *RSS*, ya que pueden utilizarse para extender la caché más allá del servidor, diciéndole al cliente (un navegador, un lector de *RSS*, etc.) cuándo puede usar una caché propia. Asimismo, tanto en el almacenamiento local, en la caché del servidor, como en la propia transmisión de los documentos *RSS*, pueden emplearse técnicas de compresión de datos que, sin afectar de manera significativa al rendimiento, reduzcan el espacio en memoria necesario para la caché así y los tiempos de transferencia entre el servidor y el cliente.


Fácil de obtener
Fácil de almacenar
Fácil de consultar



Programando en Java la Web Semántica con Jena (y IV)

Dr. DIEGO LZ. DE IPIÑA GZ. DE ARTAZA (Profesor del Departamento de Ingeniería del Software de la Facultad de Ingeniería de la Universidad de Deusto - ESIDE)

En esta cuarta y última entrega revisamos cómo la Web 2.0 actual puede beneficiarse de la adopción de ciertas tecnologías semánticas para dar lugar a portales web más inteligentes. Como muestra de ello elaboraremos un mash-up semántico capaz de agregar información heterogénea sobre eventos proveniente de diferentes organizaciones y de mostrarla de manera conjunta sobre un mapa de GoogleMaps.

Introducción

Concluimos la anterior entrega sugiriendo que la combinación de las tecnologías de microformatos, RDFa y GRDDL nos debería ayudar a progresar hacia la visión de una web con mucho más significado, explotable por las máquinas, no sólo los humanos, sin restringir y dificultar la manera en que creamos contenido web en demasía. Es lo que denominamos como "web semántica con minúsculas". La misión de esta entrega es poner los principios de la "web semántica en minúsculas" en la práctica con el desarrollo de un mash-up semántico. Mediante él, demostraremos las ventajas no solamente de esta visión más simplista de la Web Semántica sino además el potencial de la combinación de la Web 2.0 con la Web Semántica.

La Web 2.0 y la Web Semántica, aunque propugnadas por comunidades aparentemente divergentes en su visión del futuro de la Web, deberían en un futuro próximo complementarse. El argumento principal para ello es que cada una individualmente puede ayudar a resolver problemas actuales del otro enfoque.

Comenzaremos este artículo discutiendo la complementariedad de estos enfoques para luego pasar a la práctica con el desarrollo de un mash-up semántico que agrega información heterogénea, provista en diferentes formatos, sobre eventos, dando lugar a un visionado de los mismos sobre una vista geográfica.

Mejorando la Web 2.0 con la Web Semántica

El principal problema de las aplicaciones Web 2.0 actuales es que a pesar de promocionar la contribución social de montones de contenido, cada portal Web 2.0 donde se recoge tal información, es un "jardín cerrado" (*walled garden*) donde sus frutos difícilmente pueden ser recogidos y combinados con los de otros portales. De ese modo, alguien que es mi amigo en LinkedIn, puede aparecer como un extraño en MySpace. Sin embargo, como hemos visto en las entregas anteriores, la Web Semántica está concebida para resolver este tipo de problemas. De hecho, publicar datos en RDF significa que otros puedan utilizarlos más fácilmente. Aunque dos aplicaciones hayan usado un vocabulario RDF diferente es posible definir mapeos entre estos modelos de datos. Podemos combinar sentencias correspondientes a diferentes vocabularios dentro de un mismo documento para describir más formalmente su contenido.

Una manera de romper los muros de los jardines de datos de los portales Web 2.0 actuales es ofrecer APIs tal como hacen Amazon, Flickr y otras muchas organizaciones. Sin embargo, existen todavía problemas con la utilización de estos datos. A menudo, estas APIs dan lugar a estructuras de datos en XML que deben ser procesadas e integradas por los consumidores. Si se utilizara RDF en las mismas sería mucho más fácil combinar las respuestas de un portal con las de otro. Como ya se ha visto en las anteriores entregas, esto puede hacerse de dos modos: a) haciendo que las sentencias RDF de dos localizaciones diferentes hagan referencia al mismo concepto (URI) o b) estableciendo correspondencias mediante OWL indicando que dos conceptos son equivalentes. Además, el hecho de utilizar lenguajes XML como salida de las APIs de servicios de portales Web 2.0 supone que los desarrolladores acoplen

Material adicional

El material complementario puede ser descargado desde nuestra web www.revistasprofesionales.com



su código a la estructura sintáctica de la información en vez de a la estructura semántica. En consecuencia, hay que desarrollar procesadores personalizados por cada tipo de respuesta XML y no existe un mecanismo de consulta estándar para consultar estas fuentes.

Por el contrario el uso de RDF para generar respuestas desde diferentes portales de datos distribuidos permite el uso del lenguaje de consultas SPARQL. Estas consultas, en un formato similar a SQL, pueden ser ejecutadas a través de peticiones HTTP GET a servidores remotos devolviendo resultados que pueden procesarse con código estándar. Los desarrolladores lo único que necesitan es conocer la estructura de un grafo RDF detrás de un portal remoto para escribir la consulta adecuada. En definitiva, la comunidad Web 2.0 debería publicar la información en formatos más reutilizables como RDF e investigar el uso de SPARQL para recuperar datos de manera remota en vez de proporcionar cada portal una API específica. La misión de esta entrega es incidir en este aspecto, ilustrando cómo realizar tal progresión.

Mejorando la Web Semántica con la Web 2.0

La Web Semántica presenta dos problemas principales para su adopción global: disponibilidad de los datos y diseño de interfaces de interacción.

Existen muy pocos mecanismos que permitan a no expertos contribuir a la Web Semántica. No podemos hacer crecer la Web Semántica copiando/pegando código de páginas web como hicimos con la Web tradicional. Sin embargo, existen muchas aplicaciones Web 2.0 que permiten a usuarios no expertos añadir contenido (posts, imágenes) sin habilidades técnicas especiales, aunque todavía existen muy pocas o casi ninguna aplicación Web 2.0 que haga lo propio para la Web Semántica. El portal de revisiones Revyu (http://revyu.com/) es una de esas excepciones, ya que permite a través de formularios la entrada de datos de usuario que generan descripciones RDF. Este es sin duda un mecanismo esencial para hacer a los usuarios no avanzados poblar la Web Semáncia.

Otro grave problema de la web semántica es cómo generar interfaces que permita a usuarios no expertos explotar tal información. La cuestión es cómo creamos interfaces que nos permitan interaccionar sobre datos de múltiples fuentes. Un ejemplo interesante son los mash-ups basados en mapas que permiten visualizar datos bastante complejos sobre mapas. Enfoques similares deberán adoptarse a la web de datos interrelacionados infinitamente que es lo que es la Web Semántica. Adoptaremos este modo de visualización para el mash-up semántico descrito a continuación.

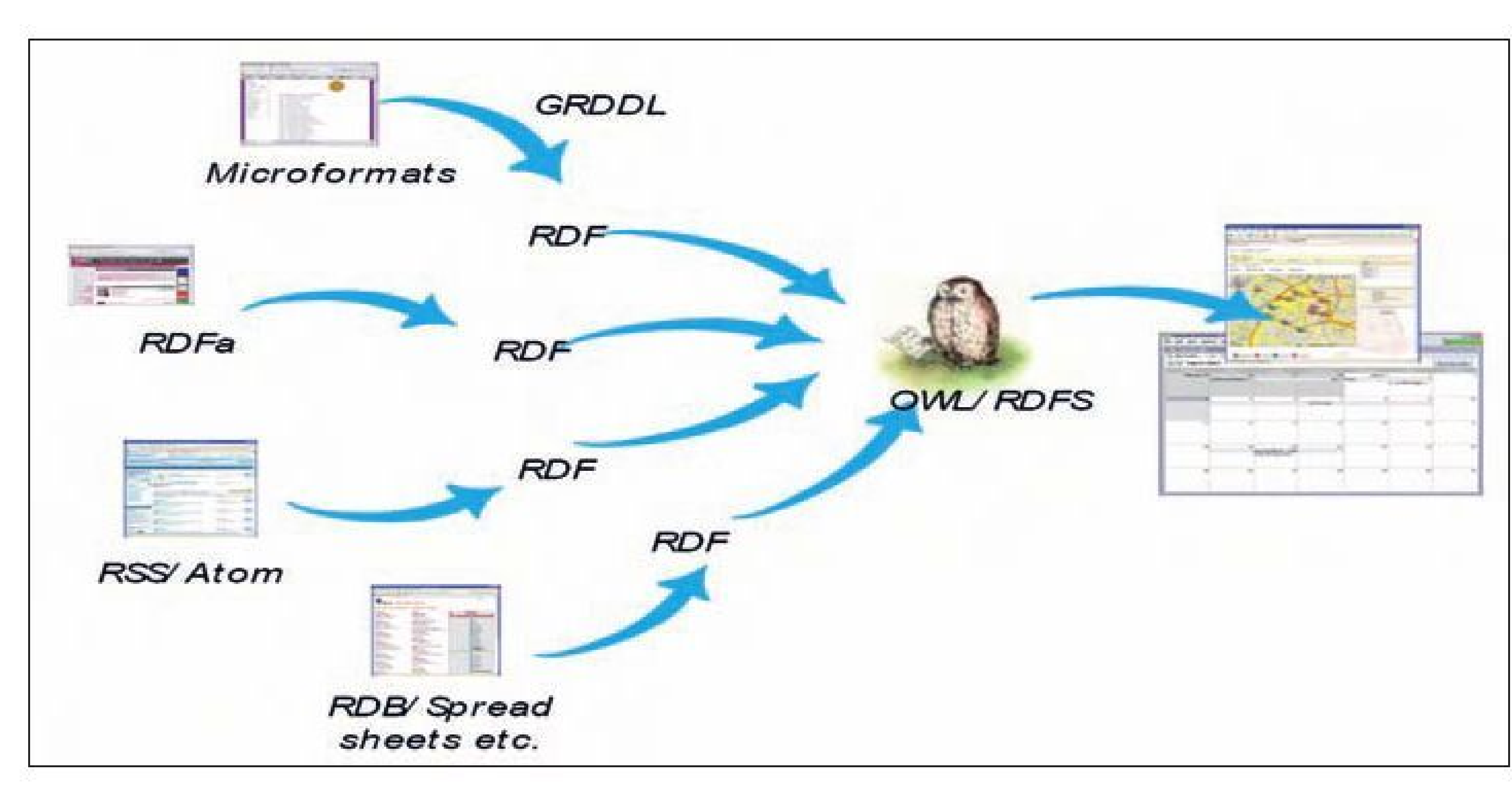


Fig.1. Arquitectura de un mash-up semántico.

En definitiva, es preciso que la comunidad de la Web Semántica dé atención urgente a la creación de interfaces que permitan a usuarios web convencionales contribuir a la Web Semántica. Esto no debería tomar la forma de editores de ontologías e instancias de RDF más amigables, sino explotar técnicas de interacción más familiares, por ejemplo, los formularios web. Además, deberían poderse desarrollar interfaces que permitan visualizar más cómodamente los grafos de relaciones entre conceptos que constituyen la Web Semántica.

Creando un mash-up semántico

Aunque lentamente, cada vez es mayor el conjunto de datos semánticos estructurados disponible en la web. Los lenguajes de Web Semántica como RDF u OWL, y los protocolos como SPARQL paulatinamente están comenzando a ser herramientas no solamente utilizadas por investigadores sino también por desarrolladores web avanzados e integradores. Esto se debe a que importantes colecciones de datos como DBLP, Wikipedia, CiteSeer o Geonames son ahora también disponibles en menor o mayor medida como conjuntos de datos RDF o puntos de consulta SPARQL. Además, portales de redes sociales como LiveJournal, Tribe o Facebook están comenzando a producir representaciones RDF de los intereses de los usuarios y la gente con quien se relacionan.

Además, hay una tendencia clara a empotrar información semántica en páginas web convencionales. De hecho, los microformatos, en particular, son cada vez más populares; Flickr ha introducido algunas etiquetas RDF y el W3C ha generado las recomendaciones RDFa y GRDDL para empotrar datos RDF en páginas XHTML ordinarias. Es sin duda, el momento para comenzar a explotar datos de estas fuentes semánticas y combinar tal información en novedosos mash-ups semánticos. En el resto del artículo describiremos los pasos necesarios para crear un mash-up que presenta sobre un mapa de GoogleMaps los eventos organizados por diferentes entidades, cuya información es publicada en formatos semánticos (RDF o RDFa) o cuasi-semánticos (microformatos).

La figura 1 muestra la arquitectura convencional de un mash-up semántico y que adoptaremos para el mash-up semántico propuesto. Obsérvese cómo información proveniente en diferentes formatos (microformatos, RDF o incluso una hoja Excel) es convertida en primer lugar a RDF y luego combinada para así ser fácilmente consultable a través de consultas SPARQL. En el caso que las fuentes de información no hagan uso de los mismos vocabularios semánticos subyacentes se requerirá un mapeo de ciertos conceptos a otros. Será necesario echar mano de RDFS u OWL para realizar esta conversión. De este modo, al final, se obtendrá un grafo en el que los conceptos modelados por diferentes vocabularios estén ligados unos con otros.

En la labor de conversión a RDF de los formatos heterogéneos suministrados por fuentes semánticas distribuidas, GRDDL juega un papel importante. GRDDL ofrece un mecanismo sencillo para la extracción de contenido RDF de dialectos XML. En vez de imponer el uso de RDF, los generadores de contenido



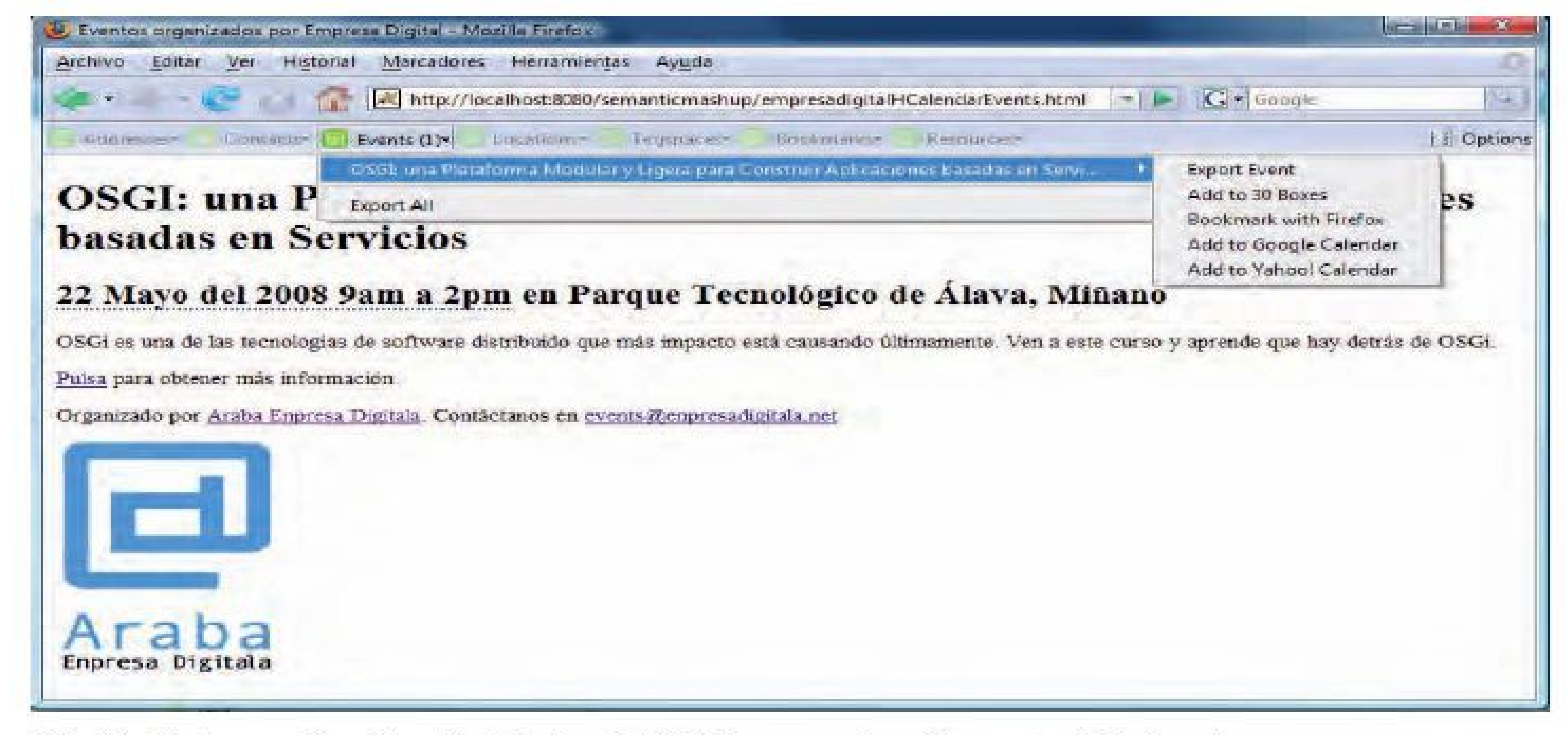


Fig.2. Vista en Mozilla de página XHTML con microformato hCalendar.

pueden utilizar un formato transformable más sencillo. GRDDL nos ayuda a asociar los algoritmos de transformación necesarios para cada dialecto, permitiendo la combinación de conceptos expresados en diferentes formatos. La framework Jena ofrece una implementación de GRDDL disponible en la página http://jena.sourceforge.net/grddl/, desde donde pueden descargarse los .jar que implementan la librería ofrecida, denominada Jena GRDDL Reader. Una vez colocados tales .jar en el CLASSPATH de tu aplicación Java, es extremadamente sencillo hacer uso de GRDDL como veremos a continuación.

Creado un agregador semántico de eventos

El mash-up a generar tiene por misión servir de agregador de descripciones de eventos publicadas por diferentes organizaciones en formatos heterogéneos convertibles a RDF. En los mash-ups Web 2.0 tradicionales, cada vez que se integra una nueva fuente de información es preciso desarrollar un nuevo adaptador que convierta los datos capturados al formato interno utilizado en el mash-up. Sin embargo, en el mash-up semántico propuesto, basta rellenar un formulario web donde se especifica la URL de publicación de eventos de un portal, así como un conjunto mínimo de metadatos (formato de los datos, descripción del portal, nombre de organización o logo de la organización) para automáticamente ser capaz de combinar los datos de este nuevo portal con los datos de otros portales antes considerados.

Los mash-up semánticos son mucho más flexibles que los mash-ups tradicionales dado que convierten la información recuperada a formato RDF, *lingua franca* que puede luego ser fácilmente luego filtrada y consultada a través de consultas SPARQL. Además, los mash-ups semánticos tienen la capacidad de evolucionar sin requerir cambios en su código. No es necesario crear un adaptador por cada nueva fuente utilizada. No obstante, aunque los datos provistos pueden ser especificados en formatos de representación sintácticos diferentes, es un requisito fundamental que tales datos, semánticamente, deben proveer una información muy similar, fácilmente convertible a un vocabulario RDF común. De hecho, las sentencias RDF obtenidas por los diferentes portales deben utilizar un vocabulario de relaciones correspondiente a una ontología de un dominio concreto. En el caso que nos ocupa tal ontología describe conceptos correspondientes a "eventos geo-localizados organizados por una entidad". Para este mash-up, en vez de diseñar una ontología desde cero, hemos preferido hacer uso de tres vocabularios RDF bien conocidos tales como Calendar, vCard y Geo, que nos permiten representar eventos con las características que deseamos.

Nuestra aplicación web de "agregación semántica de eventos geolocalizados" ofrecerá la siguiente funcionalidad:

- Permitirá añadir nuevas fuentes de eventos, indicando tanto la URL de la fuente como el formato en el que tales datos vendrán dados, de manera dinámica, sin tener que desarrollar adaptadores de fuentes de eventos específicas y recompilar el código.
- Soportará formatos sintácticos heterogéneos pero semánticamente equivalentes para la expresión de eventos. Concretamente permitirá el uso de los siguientes formatos:
- hCalendar (http://microformats.org/ wiki/hcalendar) – microformato de eventos de calendario abierto basado en el estándar iCalendar (RFC2445 (http://www.ietf.org/rfc/rfc2445.txt), muy adecuado para empotrar en HTML, XHTML, Atom, RSS o XML arbitrario.
- RDF Calendar (http://www.w3.org/TR/2005/NOTE-rdfcal-20050929/) formato RDF equivalente al estándar para la declaración de eventos de calendario iCalendar (http://www.w3.org/2002/12/ cal/rfc2445).
- El uso de expresiones RDF Calendar dentro de páginas XHTML mediante RDFa.

Enlazará los eventos descritos a través de la propiedad organizer del concepto Vevent con las entidades organizadoras de los mismos usando expresiones en microformato hCard o el formato vCard-RDF:

- vCard-RDF (http://www.w3.org/TR/vcard -rdf) – formato RDF correspondiente al perfil de tarjetas de negocio electrónicas vCard definidas por RFC 2426 (ftp://ftp.isi.edu/in-notes/rfc2426.txt).
- hCard (http://microformats.org/wiki/hcard) es un microformato para representar gente, empresas, organizaciones y lugares utilizando una correspondencia 1:1 con las propiedades y valores de vCard dentro de documentos XML.

Asociará los eventos descritos con una descripción exacta de la localización geodésica (latitud y longitud) donde se celebrarán. Para ello utilizará la propiedad geo del microformato hCalendar o expresiones en el vocabulario RDF geo en el caso de los eventos publicados en formato RDF o RDFa:

- Geo RDF (http://www.w3.org/2003/01/geo/) – simple vocabulario que permite representar información de localización en RDF. Permite encapsular la latitud y longitud de un concepto expresado mediante otro vocabulario RDF.
- Filtrará los eventos a mostrar sobre un mapa de GoogleMaps por fecha de inicio y fin y localización geográfica.

Los listados 1, 2, y 3 muestran los tres diferentes modos de representación de eventos soportados por nuestro agregador semántico. El listado 1 muestra una representación de eventos basada en microformatos, supuestamente ofrecida por la organización Empresa Digital, con sede geográfica en el Parque Tecnológico de Miñano en Álava. La representación de esta página en Mozilla Firefox puede verse en la figura 2, donde el microformato del evento ha sido detectado por la extensión Operator de Mozilla. Recordemos que como se describió en la anterior entrega, esta extensión nos permite añadir fácilmente tal evento a nuestra aplicación gestora de eventos preferida.

El listado 2 muestra parte de la descripción



LISTADO 1 — Evento geolocalizado expresado en hCalendar

```
<div id="empresadigital: event1" class="vevent">
  <h1><span class="summary">OSGI: una Plataforma Modular y Ligera para Construir Aplicaciones basadas en
Servicios</span></h1>
  <h2><abbr title="20080401T0900" class="dtstart">22 Mayo del 2008 9am a </abbr><abbr title="20080401T1400"
class="dtend">2pm</abbr> en <span class="location">Parque Tecnológico de Álava, Miñano</span></h2>
  <abbr class="geo" title="42.883; -2.760"></abbr>
  <p class="description">OSGi es una de las tecnologías de software distribuido que más impacto está causando últimamente.
Ven a este curso y aprende que hay detrás de OSGi.<a href="http://www.empresadigital.net/osgi" class="url">Pulsa</a> para
obtener más información.</p>
  <p><span class="organizer vcard">Organizado por</span>
    <a href="http://www.arabadigitala.com" class="url"><span class="fn org">Araba Enpresa Digitala</span></a>. Contáctanos
en <a class="email" href="mailto:events@enpresadigitala.net"> events@enpresadigitala.net</a>
  </p>
  <p><span class="logo" content="images/logo_ARABA.gif"> <img src="images/logo_ARABA.gif"/></span></p>
</div>
```

LISTADO 2 — Eventos geolocalizados expresado en XHMTL+RDFa

```
<?xml version="1.0" encoding="iso-8859-1"?>
<html xmlns:cal="http://www.w3.org/2002/12/cal/icaltzd#"
  xmlns:contact="http://www.w3.org/2001/vcard-rdf/3.0#"
  xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#">
<head><title>Eventos de la Facultad de Ingenieria de la Universidad de Deusto</title></head>
<body>
  <h1>Charlas organizadas por la Universidad de Deusto</h1>
  <h2 about="http://www.deusto.es/events/event1" instanceof="cal:Vevent">
    Charla técnica:<span property="cal:summary">Web con minúsculas</span></h2>
  <p><span about="http://www.deusto.es/events/event1" property="cal:description">El papel de las tecnologías GRDDL, RDFa y
los microformatos en la creación de mash-ups semánticos</span></p>
  <p>Fecha de celebración: <span about="http://www.deusto.es/events/event1" property="cal:dtstart" content="20080412T1600-
0500">12 de Marzo a las 4pm.</span> en Auditorio Principal de la Universidad de Deusto
  <span href="#p1" rel="geo:Point">
    <span about="#p1"><span property="geo:lat">43.270</span><span property="geo:long">-2.939</span></span></span>
  </p><p>Organizado por <span about="http://www.deusto.es/events/event1" href="http://www.deusto.es" rel="cal:">Universidad
de Deusto</span></p>
...
<p class="contactinfo" about="http://www.deusto.es">
  <span property="contact:fn">Universidad de Deusto</span> en <a rel="contact:url"
href="http://www.deusto.es">http://www.deusto.es</a>. Puedes contactarme en <span
property="contact:email">events@deusto.es</span> via email </p>
  <span about="http://www.deusto.es" property="contact:LOGO" content="images/deusto.jpg"><img
src="images/deusto.jpg"/></span>
</body>
</html>
```

mediante XHMTL+RDFa de un conjunto de eventos organizados supuestamente por la Universidad de Deusto.

El contenido completo de este fichero puede encontrarse en el CD de la revista. Su vista en un navegador puede comprobarse en la figura 3. Finalmente, el listado 3 muestra un fragmento del fichero RDF de descripción de eventos ofrecido supuestamente por la organización Revistas Profesionales S.L. Obsérvese en el elemento rdf:RDF la declaración de los espacios de nombres para los vocabularios vCard, Calendar y geo antes comentados.

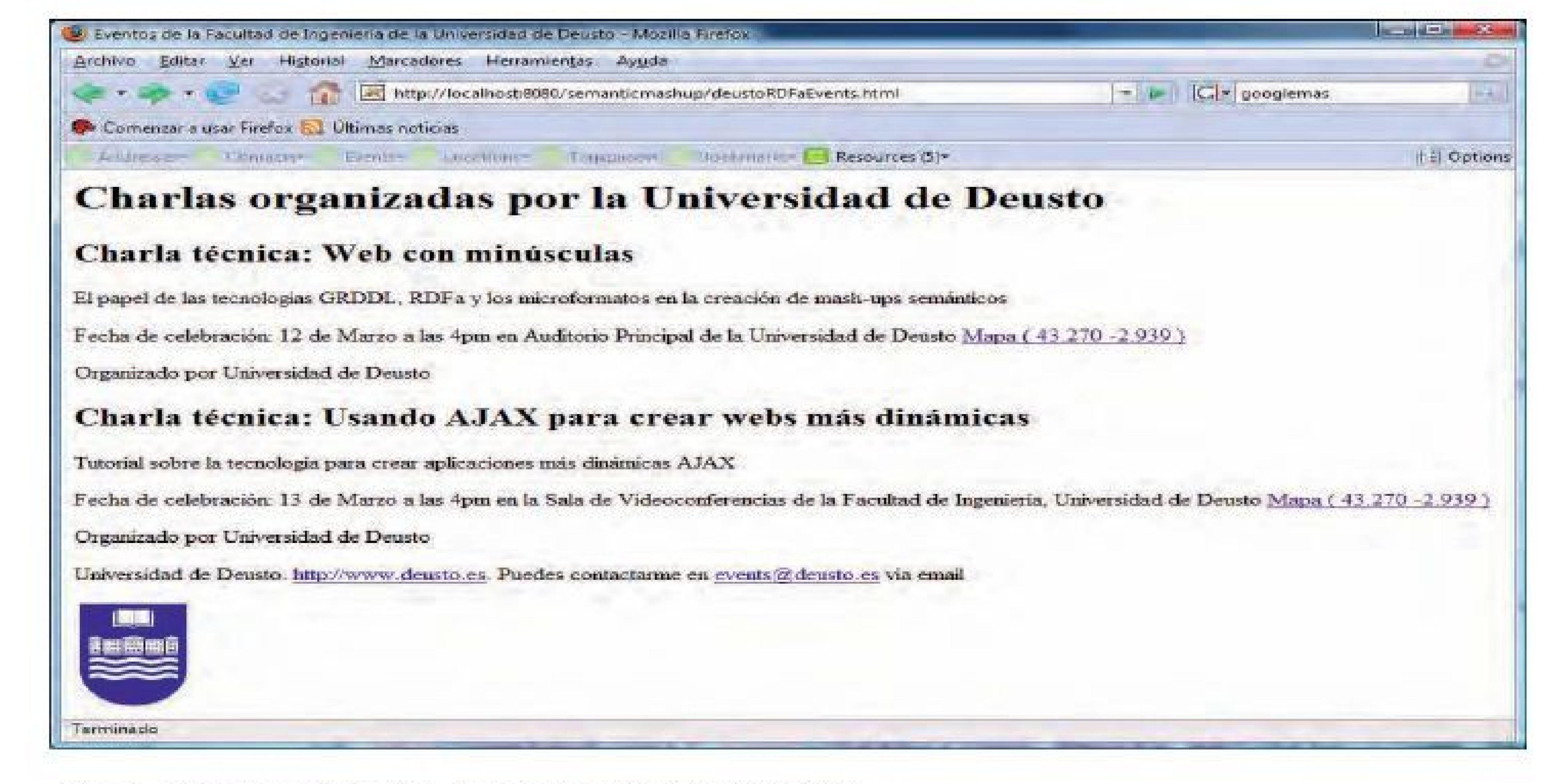
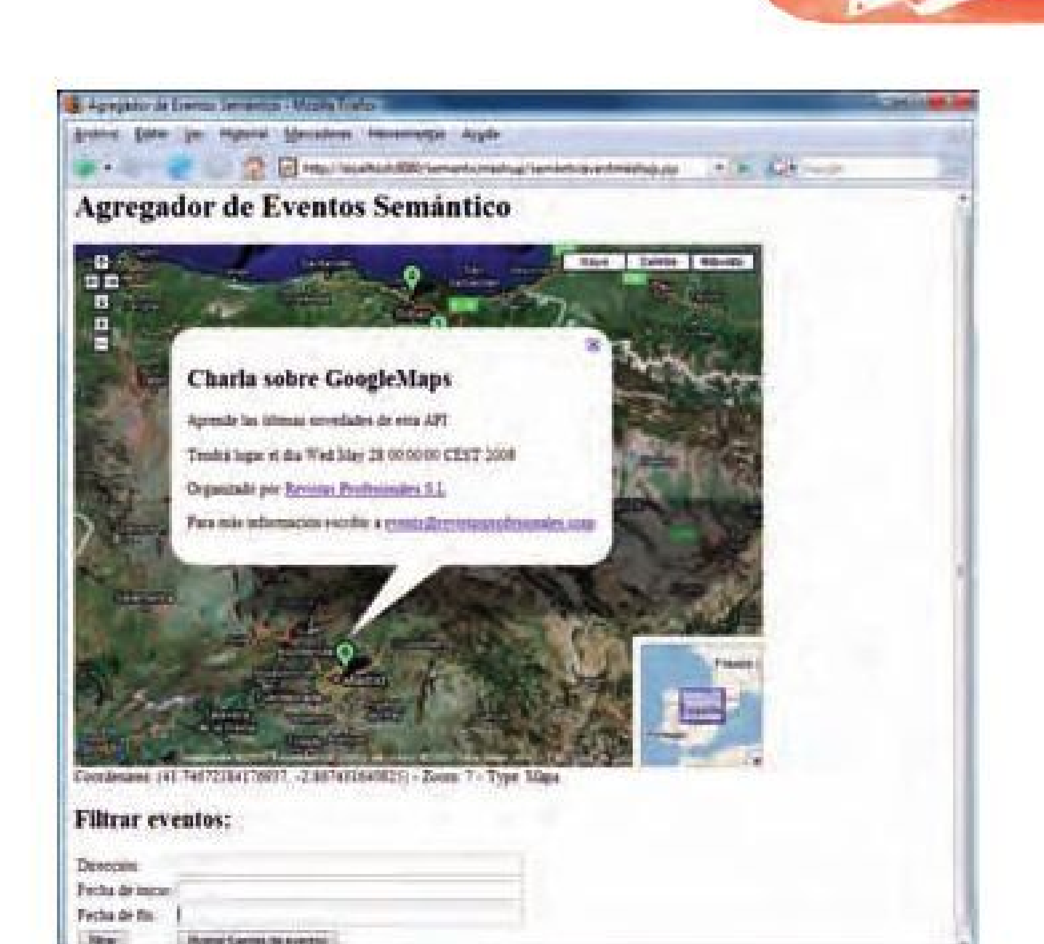


Fig.3. Vista en Mozilla de página XHTML+RDFa.

Combinando eventos con mapas

Como hemos comentado la misión de este mash-up semántico es agregar información obtenida de diversas organizaciones promotoras de eventos especificada en uno de los tres formatos mostrados en los listados 1, 2 y 3 y combinarlos con la información cartográfica de Google Maps. Analizaremos el código necesario para tratar la información semántica en la siguiente sección. Ahora nos concentramos en analizar cómo combinar el resultado del procesamiento semántico realizado por nuestro agregador de eventos, manifestado en forma de un fichero XML describiendo eventos geolocalizados, con la información cartográfica ofrecida por la API Google Maps (http://code.google.com/apis/maps/).



Vista principal de mash-up semántico agregador de eventos.

El listado 4 muestra un fragmento del fichero XML resultante de la agregación semántica de eventos de fuentes diversas. Como veremos a continuación, este fichero es muy fácilmente procesable por la API de Google Maps para generar un conjunto de marcadores en las localizaciones geodésicas donde van a tener lugar los eventos y que ofrecen una ventana informativa de los mismos al pulsar sobre ellos.

El listado 5 muestra un fragmento del fichero semanticeventmashup.jsp, alojado en el servidor de aplicaciones Tomcat, encargado de procesar el fichero XML mostrado en el listado 4 y de pintar marcadores sobre un mapa de GoogleMaps. El fichero XML es generado por el JSP RetrieveEventData.jsp cuyo código es mostrado en el listado 6. Obsérvese en el listado 5 las invocaciones a las clases GMap2 para generar un mapa, GDownloadUrl para acceder a un recurso web remoto y GXml para procesar un fichero XML usando DOM, ofrecidos por la API de GoogleMaps. Omitimos el uso de la clase GMarker utilizada para generar los marcadores y los pop-ups informativos sobre cada evento ofrecido por nuestro mash-up semántico, mostrado en la figura 4. El código completo de este .jsp puede revisarse en el CD de la revista. Obsérvese además que este JSP es responsable de presentar un formulario a través del cual se pueden filtrar los eventos presentados en el mapa por localización, fecha de inicio y fecha de fin, revisar parte inferior de la figura 4.

Usando Jena para agregar eventos

La clase principal de nuestro agregador semántico es es.solop.semanticmashup.EventAggregator. Realiza las siguientes tres funciones básicas:

- Permite el registro en tiempo de ejecución de nuevas fuentes semánticas de eventos, recuperando su contenido.
- Fusiona las fuentes semánticas con descripciones de eventos en formato RDF, RDFa y hCalendar que son registradas con el mashup y mapea los conceptos representados por ellas.
- Ejecuta una consulta SPARQL sobre el modelo fusionado parametrizada por rangos de fecha que devuelve el conjunto de eventos geolocalizados en curso en las fechas indicadas.

El listado 7 muestra los tres métodos utilizados por la clase EventAggregator para obtener RDF de fuentes distribuidas que suministran información sobre eventos en los siguientes formatos semánticos: a) el convencional RDF, b) el formato RDFa, ideal para empotrar RDF en páginas XHTML y c) el microformato hCalendar.

El método getEventModelFromRDF simplemente hace uso del método Model Maker.createModel para crear un modelo en memoria en el que se carga el contenido RDF/XML de una URL pasada como parámetro.

El método getEventModelFromRDFa se encarga de procesar el contenido de una página en formato XHTML+RDFa y de cargar el contenido RDF obtenido a través de una transformación GRDDL en el modelo declarado. Dado que Jena tiene implícitamente soporte para extraer las sentencias RDF en un fichero XHTML+RDFa, no

LISTADO 3 — Eventos geolocalizados expresado en RDF

```
<?xml version="1.0" encoding="iso-8859-1"?>
<rdf:RDF xmlns:cal=http://www.w3.org/2002/12/cal/icaltzd#
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
  xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#">
  <rdf:Description rdf:about="http://www.revistasprofesionales.com/solop#event2">
    <rdf:type rdf:resource="http://www.w3.org/2002/12/cal/icaltzd#Vevent"/>
    <cal:organizer rdf:resource="http://www.revistasprofesionales.com/solop"/>
    <cal:dtend rdf:datatype="http://www.w3.org/2001/XMLSchema#date">2008-05-27</cal:dtend>
    <cal:summary rdf:datatype="http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral">Charla sobre Web 3.0</cal:summary>
    <cal:description rdf:datatype="http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral">Ven a ver lo último en la siner-
gia entre Web 2.0 y Web Semántica (GRDDL, RDFa, microformátos)</cal:description>
    <cal:url rdf:resource="http://www.revistasprofesionales.com/solop/charla2"/>
    <cal:dtstart rdf:datatype="http://www.w3.org/2001/XMLSchema#date">2008-05-27</cal:dtstart>
    <cal:location rdf:datatype="http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral">Madrid, España</cal:location>
      <geo:Point>
        <rdf:Description rdf:about="p1">
          <geo:lat rdf:parseType="Literal">40.437</geo:lat>
            <geo:long rdf:parseType="Literal">-3.625</geo:long>
        </rdf:Description>
      </geo:Point>
  </rdf:Description>
...
  <rdf:Description rdf:about="http://www.revistasprofesionales.com/solop">
    <rdf:type rdf:resource="http://www.w3.org/2001/vcard-rdf/3.0#vCard"/>
      <vcard:fn>Revistas Profesionales S.L</vcard:fn>
    <vcard:url rdf:resource="http://www.revistasprofesionales.com/solop"/>
    <vcard:email>events@revistasprofesionales.com</vcard:email>
      <vcard:logo vcard:TYPE="image/png"
                  rdf:resource="images/solop.png"/>
      <vcard:adr rdf:parseType="Resource">
              <vcard:Street>   C/ Valentín Beato 42, 3ª Planta </vcard:Street>
              <vcard:Locality> Madrid </vcard:Locality>
              <vcard:Pcode>     28037 </vcard:Pcode>
              <vcard:Country>  España </vcard:Country>
      </vcard:adr>
  </rdf:Description>
</rdf:RDF>
```

LISTADO 4 — Fichero XML descriptor de eventos geolocalizados

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<markers>
<marker summary="OSGI: una Plataforma Modular y Ligera para Construir Aplicaciones basadas en Servicios" description="OSGi
es una de las tecnologías de software distribuido que más impacto está causando últimamente. Ven a este curso y aprende
que hay detrás de OSGi." startDate="Tue Apr 01 09:00:00 CEST 2008" endDate="Tue Apr 01 14:00:00 CEST 2008" latitu-
de="42.909794" longitude="-2.66515" orgName="Araba Enpresa Digitala" orgUrl="http://www.arabadigitala.com"
orgEmail="events@enpresadigitala.net" orgLogo="images/logo_ARABA.gif"/>
...
<marker summary="Web con minúsculas" description="El papel de las tecnologías GRDDL, RDFa y los microformatos en la crea-
ción de mash-ups semánticos" startDate="Sat Apr 12 16:00:00 CEST 2008" endDate="null" latitude="43.27" longitude="-2.939"
orgName="Universidad de Deusto" orgUrl="http://www.deusto.es" orgEmail="events@deusto.es" orgLogo="images/deusto.jpg"/>
...
<marker summary="Charla sobre Web Semantica" description="Tutorial sobre las tecnologías base de la Web Semántica: RDF,
RDFS, OWL y SPARQL" startDate="Sat Mar 29 00:00:00 CET 2008" endDate="null" latitude="40.437" longitude="-3.625"
orgName="Revistas Profesionales S.L" orgUrl="http://www.revistasprofesionales.com/solop" orgEmail="events@revistasprofesio-
nales.com" orgLogo="images/solop.png"/>
</markers>
```

LISTADO 5 — Front-end web del mashup:

```
<%@page contentType="text/html" %>
<%@page import="java.util.*, es.solop.semanticmashup.*"%>
<jsp:useBean id="eventAggregatorBean" class="es.solop.semanticmashup.EventAggregator" scope="application"/>
<html xmlns="http://www.w3.org/1999/xhtml"encoding="iso-8859-1">
  <head>
    <title>Agregador de Eventos Sem&aacute;ntico</title>
    <script src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=..."
      type="text/javascript">//<![CDATA[
       ...
      function load() {
        ...
        map = new GMap2(document.getElementById("map"));
        ...
        map.addControl(new GSmallMapControl());
        map.addControl(new GMapTypeControl());
        map.addControl(new GOverviewMapControl());
        map.setCenter(initialPoint, initialZoom);
        map.setMapType(G_HYBRID_MAP);
        GDownloadUrl("http://localhost:8080/semanticmashup/RetrieveEventData.jsp", function(data, responseCode) {
          var xml = GXml.parse(data);
          var markers = xml.documentElement.getElementsByTagName("marker");
          for (var i = 0; i < markers.length; i++) {
            var bodyMarker = "<h2>" + markers[i].getAttribute("summary") + "</h2>";
            bodyMarker += "<p>" + markers[i].getAttribute("description") + "</p>";
            bodyMarker += "<p> Tendr&aacute; lugar el d&iacute;a " + markers[i].getAttribute("startDate") + "</p>";
            bodyMarker += "<p> Organizado por <a href=\"" + markers[i].getAttribute("orgUrl") + "\">" +
markers[i].getAttribute("orgName") + "</a></p>";
            bodyMarker += "<p> Para m&aacute;s informaci&oacute;n escribir a <a href=\"mailto:" +
markers[i].getAttribute("orgEmail") + "\">" + markers[i].getAttribute("orgEmail") + "</a></p>";
           var point = new GLatLng(parseFloat((markers[i].getAttribute("latitude")) + i),
                                   parseFloat((markers[i].getAttribute("longitude"))+i));
            map.addOverlay(createMarker(point, i+1, "green", bodyMarker));
        }});
      }
    }//]]>
    </script>
  </head>
  <body onload="load()" onunload="GUnload()">
    <h1>Agregador de Eventos Sem&aacute;ntico</h1>
      <div id="map" style="width: 800px; height: 600px"></div>
      ...
      <h2>Filtrar eventos:</h2>
      <form action="/semanticmashup/semanticeventmashup.jsp?action=filter">
        ...
       </form>
       // Código para añadir nuevas fuentes de eventos
      ...
  </body>
</html>
```

es necesario indicar ninguna declaración del algoritmo de transformación en el documento original. Basta con obtener un com.hp.hpl.jena.rdf.model.RDF Reader para GRDDL del modelo y asociarle la propiedad grddl.rdfa a true. Al pasar una URL a tal instancia de RDFReader se recuperará y transformará su contenido a RDF, de manera transparente.

Finalmente, el método getEventModelFrom HCalendar sí que usa un algoritmo de conversión explicitado en los propios documentos XHTML con microformato hCalendar para a través de la clase RDFReader efectuar la conversión necesaria y cargar el modelo RDF. Las siguientes líneas deben colocarse en todo documento en



formato XHTML+hCalendar conteniendo descripciones de eventos que quieran ser interpretados por nuestro agregador de eventos semánticos. La URL http://www. w3.org/2002/12/cal/glean-hcal apunta a una hoja de estilo XSLT que transforma el contenido XHTML+hCalendar a RDF.

```
<head profile="http://www.w3.org/2003/
  g/data-view">
  ...
  <link rel="transformation"
href="http://www.w3.org/2002/12/cal/
glean-hcal"/>
</head>
```

El listado 8 muestra cómo los diferentes modelos RDF recuperados son fusionados a través del método union()de la clase com.hp.hpl.jena.rdf.model.Model. Por otro lado, el listado 9 muestra el método EventAggregator.query EventModel encargado de ejecutar consultas SPARQL sobre el modelo RDF unificado conteniendo descripciones de eventos provenientes de diferentes fuentes. Lo más reseñable de este método es el uso de las clases QueryFactory, QueryExecution y ResultSet del paquete com.hp.hpl.jena.query para realizar la consulta del modelo y obtener como resultado una lista de objetos de tipo GeoEvent. Tal clase recoge toda la información necesaria para poder visualizar información sobre un evento geolocalizado organizado por alguien, es decir, recoge los detalles del evento (descripción, resumen, url), su localización (latitud y longitud) y los detalles del organizador (nombre, url, email y logo). La clase es.solop.semanticmashup.GeoEvent puede encontrarse con el resto del código del mash-up en el CD de la revista. Finalmente, el listado 10 muestra la consulta SPARQL utilizada para extraer del grafo combinado de todas las fuentes de eventos semánticas, los detalles de los eventos que presentaremos sobre un mapa de Google Maps. Tales detalles obtenidos como una lista de instancias de GeoEvent son transformados por el método EventAggregator.get EventMetadataXML (revisar el código en el CD) en un fichero XML tal como el que mostramos en el listado 4. Algo importante referente a la consulta SPARQL del listado 10 es que permite el filtrado por fecha de inicio y final de los eventos. Tal capacidad de filtrado es explotada en nuestro mash-up cada vez que se pulsa sobre el botón "Filtrar" de la figura 4. Nótese también que tal consulta soporta dos modos alternativos de representar la latitud y longitud geodésica de un evento. Por esa razón hay dos bloques OPTIONAL referentes a información geodésica.

LISTADO 6 — RetrieveEventData.jsp

```
<%@page session="false"%>
<jsp:useBean id="eventAggregatorBean"
class="es.solop.semanticmashup.EventAggregator" scope="application"/>
<% out.print(eventAggregatorBean.getEventMetadataXML()); %>
```

LISTADO 7 — Métodos de recuperación RDF de EventAggregator

```
public Model getEventModelFromRDF(String url) {
   Model rdfModel =
ModelFactory.createMemModelMaker().createModel("modelFromRDF",false);
   rdfModel.read(url,"RDF/XML");
   return rdfModel;
}
public Model getEventModelFromRDFa(String url) {
   Model m = ModelFactory.createMemModelMaker().createModel("modelFromRDFa");
   RDFReader r = m.getReader("GRDDL");
   r.setProperty("grddl.rdfa", "true");
   r.read(m, url);
   return m;
}
public Model getEventModelFromHCalendar(String url) {
   Model m = ModelFactory.createMemModelMaker().createModel("modelFromRDFa-1");
   RDFReader r = m.getReader("GRDDL");
   r.read(m, url);
   return m;
}
```

LISTADO 8 — Fusionado de fuentes de información

```
private Model mergeModels() {
   Model m = ModelFactory.createMemModelMaker().createModel("mergedModel");
   for (EventModelMetadata model: this.models) {
      Model tempModel =
ModelFactory.createMemModelMaker().createModel("tempModel");
      switch (model.getType()) {
         case RDFa:
            tempModel = this.getEventModelFromRDFa(model.getUrl());
            break;
         case hCalendar:
            tempModel = this.getEventModelFromHCalendar(model.getUrl());
            break;
         case RDF:
            tempModel = this.getEventModelFromRDF(model.getUrl());
            break;
      }
      m = m.union(tempModel);
   }
   return m;
}
```

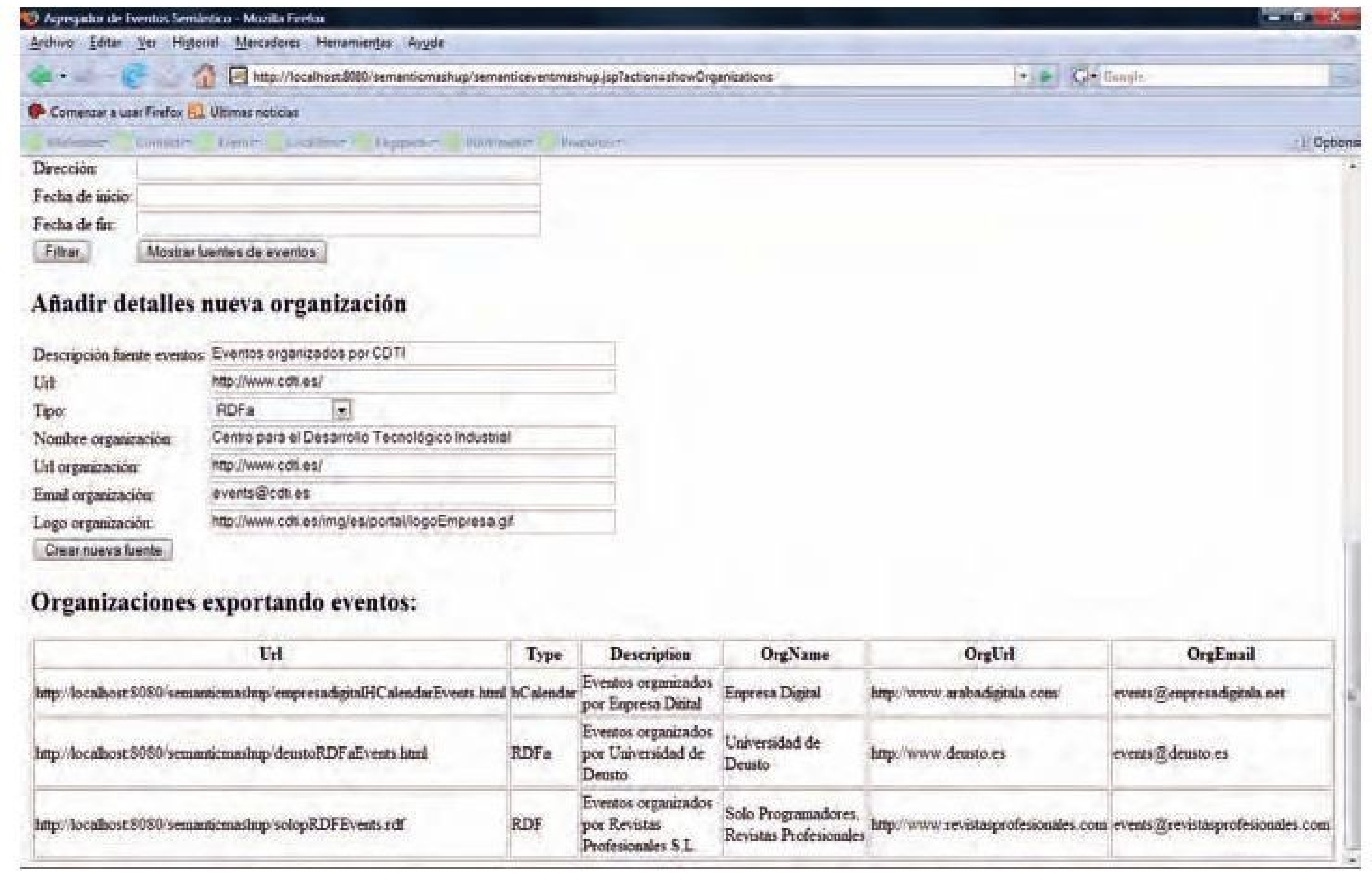


Fig.5. Formulario para añadir nueva fuente de eventos semántica.



LISTADO 9 — Consultando el modelo de eventos RDF

```
public List<GeoEvent> queryEventModel(Model m, String queryStr) {
  Query query = QueryFactory.create(queryStr);
  QueryExecution qe = QueryExecutionFactory.create(query, m);
  ResultSet results = qe.execSelect();
  List<GeoEvent> events = new ArrayList<GeoEvent>();
  for(; results.hasNext();) {
    QuerySolution solution = results.nextSolution();
    Literal summary = solution.getLiteral("summary");
    String summaryStr = summary == null ? "": summary.getString().trim();
    ...
    Literal latitude = solution.getLiteral("lat");
    float latitudeFloat = latitude == null ? 0:
Float.parseFloat(latitude.getString());
    events.add(new GeoEvent(summaryStr, descriptionStr, startDateStr,
endDateStr, latitudeFloat, longitudeFloat, orgNameStr, orgUrlStr, orgEmailStr,
orgLogoStr));
  }
  qe.close();
  return events;
}
```

LISTADO 10 — Consultando el modelo de eventos con SPARQL

```
PREFIX cal: <http://www.w3.org/2002/12/cal/icaltzd#>
PREFIX contact: <http://www.w3.org/2001/vcard-rdf/3.0#>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX xsd:    <http://www.w3.org/2001/XMLSchema#>
SELECT DISTINCT ?summary ?description ?dtstart ?dtend ?lat ?long ?org ?url
?email ?logo
WHERE {
  ?event cal:summary ?summary.
  ?event cal:dtstart ?dtstart.
  OPTIONAL { ?event cal:organizer ?organizer.
             ?organizer contact:fn ?org.
             ?organizer contact:url ?url.
             ?organizer contact:email ?email.
             OPTIONAL { ?organizer contact:logo ?logo. } }
  OPTIONAL { ?event geo:Point ?point.
             ?point geo:lat ?lat.
             ?point geo:long ?long. }
  OPTIONAL { ?event cal:geo ?loc.
             ?loc <http://www.w3.org/1999/02/22-rdf-syntax-ns#first> ?lat.
             ?loc <http://www.w3.org/1999/02/22-rdf-syntax-ns#rest> ?restgeo.
             ?restgeo <http://www.w3.org/1999/02/22-rdf-syntax-ns#first> ?long.
}
  OPTIONAL { ?event cal:dtend ?dtend. }
  OPTIONAL { ?event cal:description ?description. }
  FILTER ( xsd:dateTime(?dtstart) >= xsd:dateTime("2008-04-14T00:00:00Z") )
  FILTER ( xsd:dateTime(?dtend) <= xsd:dateTime("2008-04-21T00:00:00Z") )
} ORDER BY ?event;
```

LISTADO 11 — Añadiendo una nueva fuente semántica de datos RDF al mashup

```
public void addModel(String eventSourceUrl, String eventDescription, String
eventSourceTypeStr, String orgName, String orgUrl, String orgEmail, String
orgLogo) {
  EventModelMetadata.EventModelType eventSourceType;
  if (eventSourceTypeStr.equals("RDF")) {
    eventSourceType = EventModelMetadata.EventModelType.RDF;
  } else if (eventSourceTypeStr.equals("RDFa")) {
    eventSourceType = EventModelMetadata.EventModelType.RDFa;
  } else {
    eventSourceType = EventModelMetadata.EventModelType.hCalendar;
  }
  this.models.add(new EventModelMetadata(eventSourceUrl, eventDescription,
eventSourceType, orgName, orgUrl, orgEmail, orgLogo));
  this.mergedModel = this.mergeModels();
}
```

El listado 11 muestra el método EventAggrega tor.addModel utilizado para alimentar al mashup agregador con nuevas fuentes semánticas de eventos. La clase es.solop.semanticmashup. EventModelMetadata es utilizada para almacenar los datos relativos a cada fuente descriptora de eventos. Tal método es invocado cada vez que en la página JSP semanticeventmashup.jsp tras pulsar sobre el botón "Mostrar fuente de eventos", se rellenan los detalles de una nueva fuente de eventos y se pulsa el botón "Crear nueva fuente" como muestra la figura 5.

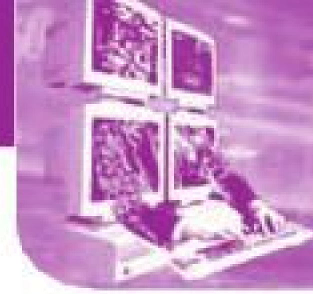
El código completo del mash-up semántico agregador de eventos puede encontrarse en forma de una aplicación web completa para Tomcat en el CD de la revista. Para su despliegue tan solo es necesario pegar la carpeta semantic-mashup del CD en la carpeta webapps de Tomcat.

Conclusión

Esta última entrega de la serie sobre programación semántica con Jena nos ha mostrado el potencial de añadir capacidades de procesamiento semántico a los actuales mash-ups Web 2.0, dando lugar a los mash-ups semánticos. Sin duda, la progresiva adopción de tecnologías de la "web semántica con minúsculas" a los portales Web 2.0 actuales, nos permitirá una mejor combinación e integración de datos provenientes de fuentes semánticas heterogéneas muy fácilmente consultables mediante SPARQL, tal como hemos revisado en esta entrega. La ventaja fundamental del mash-up semántico descrito frente a los mash-ups convencionales es que permite su continua evolución, sin necesidad de tener que desarrollar nuevos adaptadores para cada nueva fuente de información tal como ocurre hasta ahora. Espero que esta serie haya servido para ir perdiendo el miedo a la Web Semántica, descubrir su potencial y alentar al lector a su uso en el desarrollo de aplicaciones web de nueva generación.

Referencias

- How to Combine the Best of Web2.0 and a Semantic Web: Examples from Revyu.com, http://kmi.open.ac.uk/people/tom/papers/heath-motta-www2007dev-combining-web20-semantic-web.pdf, Tom Heath y Enrico Motta, Developers' Track, 16th International World Wide Web Conference (WWW2007), Banff, Canada, 2007
- GRDDL Data Views: Getting Started, Learning More, http://www.w3.org/2003/g/data-view
- RDFa Primer: Embedding Structured Data in Web Pages, http://www.w3.org/ TR/xhtml-rdfa-primer/, 2008
- Especificación Microformatos hCard, hCal y Geo – http://microformats.org /wiki/hcard, http://microformats.org/ wiki/hCalendar y http://microformats. org/wiki/geo.
- Semantic Mash- ups using RDF, RSS and Microformats, Dean Allemang, JAZOON07, Junio 2007, http://jazoon.com/ download/presentations/870.pdf.



Scrum (I) Desarrollando ágilmente: las responsabilidades

JAVIER HOLGUERA BLANCO - javier.holguera@gmail.com
Titulado en Ingeniería en Informática,
Analista/Programador en GMV-SGI.

En las últimas décadas la ingeniería del software ha ofrecido una gran variedad de soluciones para lograr una finalización satisfactoria de los proyectos informáticos. Se ha pasado del paradigma estructurado al orientado a objetos, de los procesos pesados a las filosofías ágiles de desarrollo.

Esta serie de dos artículos va a tratar sobre Scrum, uno de los resultados de esta filosofía ágil que más éxito ha tenido y que, en los últimos tiempos, está empezando a hacerse un hueco entre las preferencias de los gestores de proyectos en España. Scrum no es la bala de plata de la que hablaba Brooks en 1986, pero es una herramienta potente que ofrece excelentes resultados a quien se decide a utilizarla.

Introducción

No es fácil dividir una explicación completa y detallada de Scrum en dos artículos, sin que el primero de ellos quede falto de ciertas descripciones, a expensas de leer el segundo para comprender en su totalidad algunos conceptos. El primer artículo se centra en la descripción de Scrum, sus participantes y los artefactos en él utilizados, buscando crear una panorámica general que sea complementada en el segundo artículo con detalles más completos sobre los eventos de Scrum.

La primera parte del presente artículo indica qué es Scrum y qué características tiene. En la segunda sección se describe brevemente un entorno real de utilización de Scrum, así como algunas de las conocidas empresas que lo están usando en muchos de sus proyectos. La tercera sección se centra en definir de qué forma se organiza el trabajo en Scrum y cuáles son las medidas de tiempo que se utilizan, para hacer estimaciones y medir la productividad lograda. El resto del artículo se enfoca en describir los distintos roles que pueden ejercer los participantes en Scrum, así como los artefactos que van a tener que manejar. Por último, como es habitual, el artículo se cierra con las correspondientes conclusiones sobre lo descrito, así como una breve reseña al contenido del siguiente.

¿Qué es Scrum?

No resulta trivial dar una definición concreta y sencilla de lo que es Scrum. Haciendo caso a su creador, Ken Schwaber, Scrum es un framework de desarrollo, más que un proceso. Mientras que los procesos tradicionales indican "cómo" se debe gestionar el trabajo correspondiente a la elaboración de un proyecto de software, en Scrum las indicaciones se limitan a "qué" se debe hacer, no entrando a dictar el "cómo".

Las reglas que propone Scrum deben ser adaptadas a cada entorno empresarial concreto, buscando mediante esta personalización exprimir sus beneficios al máximo. El resultado de adaptar estas reglas del framework Scrum, es un proceso en el sentido habitual del término, puesto que se ha pasado del conocimiento del "qué", inicialmente expuesto por Scrum, al "cómo" propio de cualquier proceso.

En el resto de este artículo y en el posterior de la saga, a la hora de referirse a Scrum como proceso, se hace referencia a cualquiera de las implementaciones concretas de Scrum, una vez adaptado.

Los procesos resultantes de la adaptación de Scrum tienen una serie de características comunes. El principal rasgo compartido es la naturaleza ágil de todos estos procesos, puesto que el propio framework es uno de los principales referentes del desarrollo ágil, junto con XP (eXtreme Programming). Gracias a dicha agilidad, estos procesos son capaces de facilitar el cambio en los requisitos del proyecto, aspecto imprescindible hoy en día para adaptar los desarrollos de software a las condiciones mutables de los distintos mercados en los que participan.

La capacidad de una empresa para hacer frente a estos cambios y asumirlos como propios, en lugar de señalar al cliente como culpable de los mismos,

Material adicional

El material complementario puede ser descargado desde nuestra web www.revistasprofesionales.com



es uno de los lemas a respetar si se quiere desarrollar aplicaciones software que satisfagan las necesidades de sus futuros usuarios.

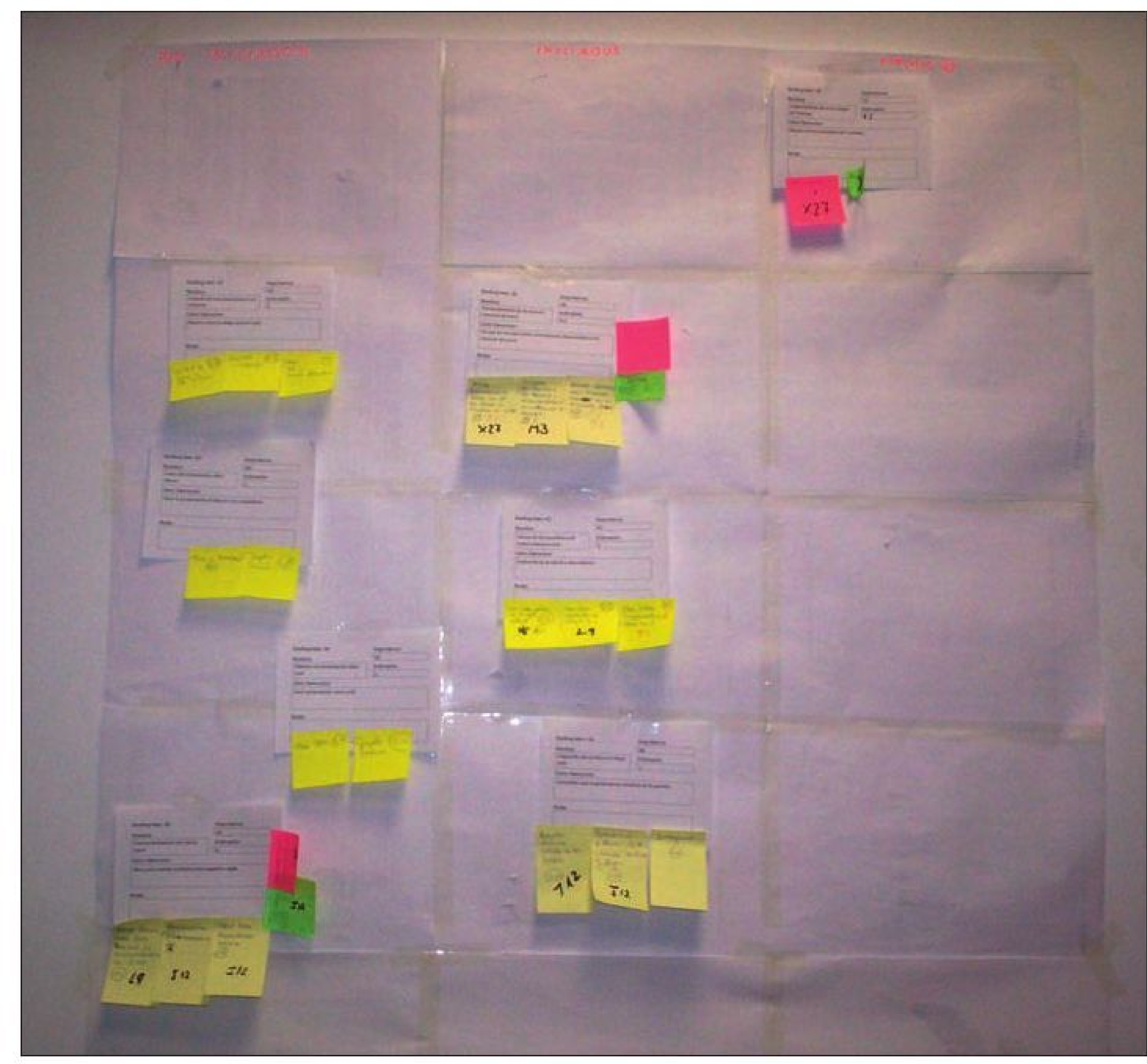
Otra característica de cualquier proceso Scrum es la iteratividad. Todo el desarrollo del proyecto se divide en una sucesión de ciclos cortos, de un par de semanas, conocidos como sprints. Éste es uno de los primeros términos con los que el lector debe tomar familiaridad al introducirse en Scrum, existiendo otros que se irán describiendo a lo largo de este y el posterior artículo y que completan el vocabulario propio del framework.

Los sprints de Scrum pueden tener una duración variable, en un rango de dos a cuatro semanas. Normalmente los gestores de proyecto suelen preferir una duración lo más corta posible, para tener resultados inmediatos, minimizar los riesgos asociados a la incertidumbre y controlar de forma más férrea el avance del proyecto. Por el contrario, los desarrolladores suelen preferir sprints de más larga duración que les permitan tener más flexibilidad a la hora de organizar su trabajo y les eviten tener que rendir cuentas sobre sus resultados con mayor frecuencia. Es recomendable probar con varias duraciones al introducir Scrum en el entorno empresarial, hasta encontrar aquella que ofrezca los resultados óptimos de productividad y satisfacción en los participantes.

Una vez alcanzada la cifra exacta, es muy conveniente mantener la duración de los sprints a dicha cantidad de semanas siempre que sea posible. La sucesión de sprints de duración predeterminada benefician a la organización de la empresa, puesto que establece una pauta que facilita a la alta dirección el seguimiento de los distintos proyectos acometidos en la misma.

Un proceso Scrum es, además de iterativo, incremental, como es habitual en los procesos modernos. Este carácter incremental en la forma de elaborar el proyecto ve su reflejo en una de las reglas de oro de Scrum: al finalizar cada uno de los sprints, es necesario presentar un incremento de funcionalidad. Este incremento debe estar totalmente terminado, listo para pasar al entorno de producción de ser necesario. Es decir, debe haber pasado todas las pruebas que se hayan definido, así como las correspondientes métricas de calidad que el proyecto haya establecido.

Con objeto de reforzar al equipo en la idea de que el trabajo que se entregue debe estar totalmente completo, Scrum propone realizar una demostración, al final de cada ciclo, del incremento desarrollado. A esta demostración puede asistir cualquiera que esté interesado en la realización del proyecto, siendo especialmente recomendable que esté presente el cliente.



Sprint Backlog físico.

En cualquier caso, tanto de éste como de los demás eventos de Scrum, se realizará una descripción mucho más pormenorizada en el siguiente artículo, indicando la utilidad de cada uno, los participantes que están autorizados a existir al evento (así como las acciones que tendrán permitido realizar) y la forma en que realizan.

Un caso real

Es interesante indicar que lo expuesto en este documento ha sido puesto en práctica en un entorno empresarial real. Las explicaciones que se dan en este y el posterior artículo sobre Scrum y sobre la forma de adaptar sus reglas, corresponden a la implantación hecha en Clay Formación Internacional, en cuyo departamento de I+D+i se ha desplegado este framework, obteniendo como resultado un proceso concreto altamente adaptado a las condiciones del mismo.

La naturaleza propia de un departamento de I+D+i, que debe ser lo más permeable posible a las nuevas tendencias y a los cambios en las líneas de investigación, hacen de Scrum ideal para los desarrollos que se realizan bajo estas condiciones. Scrum aporta la flexibilidad que el departamento de Clay demandaba para hacer aplicaciones innovadoras.

En cualquier caso, las indicaciones que se hacen en este artículo sobre cómo desplegar y adaptar Scrum, no se deben tomar como una verdad absoluta. Si algo intentan transmitir los autores más relevantes relacionados con Scrum, entre ellos el propio Ken Schwaber como creador del mismo, es que en cada empresa, inclusive en los distintos departamentos de una misma empresa, la implantación de Scrum puede responder a reglas distintas y lograrse de formas muy diferentes.

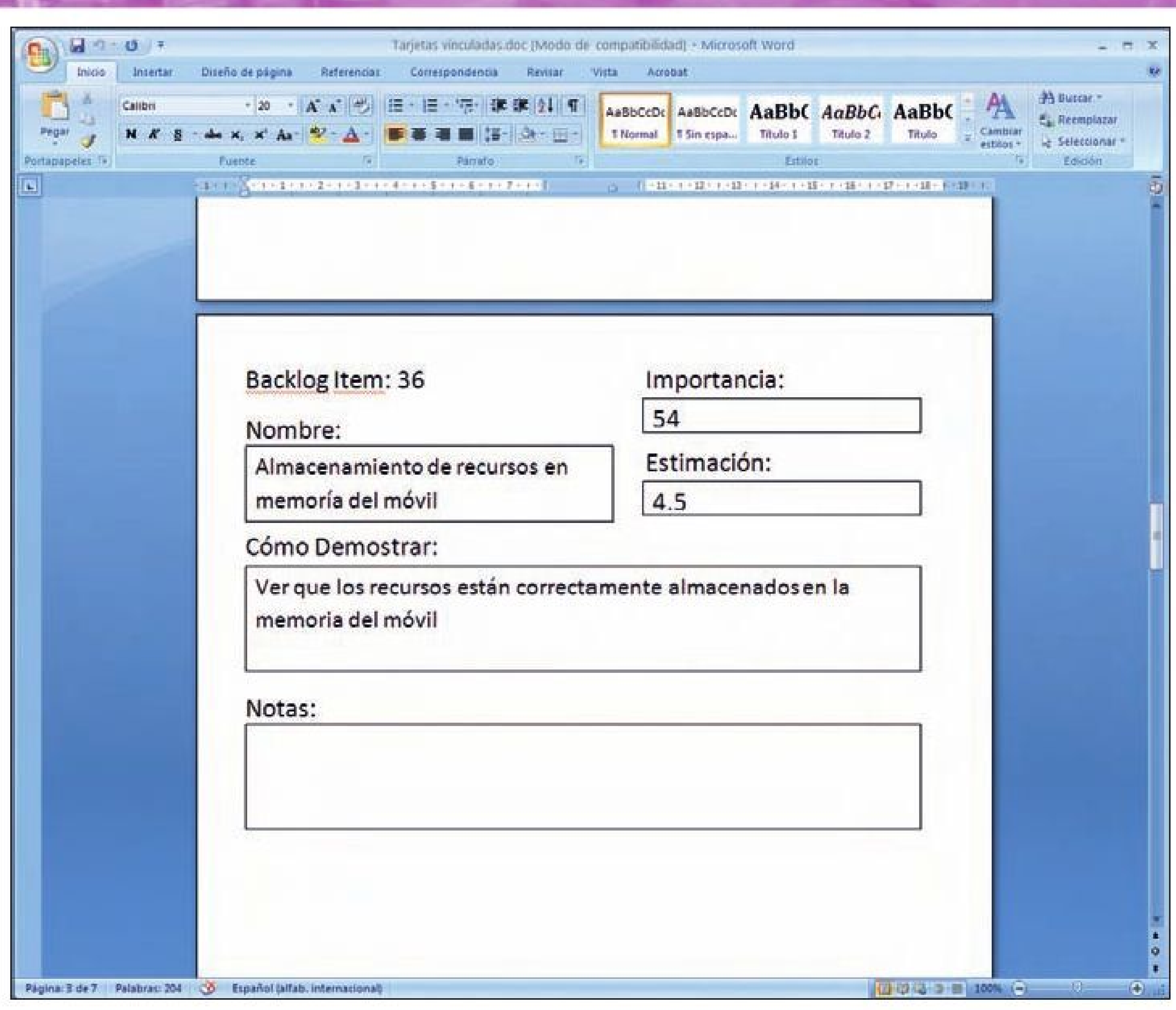
Es responsabilidad de aquellos que se encarguen de realizar dicha implantación y de quienes, posteriormente, se ocupen de mantener el proceso en "plena forma", buscar la mejor adaptación posible del framework al entorno en el que se quiere implantar. Uno de los principios de las filosofías ágiles de desarrollo de software es que son los procesos los que se deben adaptar a las personas, y no al revés, puesto que son las personas el principal activo con que cuenta la empresa. Cuanto mayor sea la satisfacción de estas personas en la



forma de trabajar, mejores serán los resultados que produzcan.
Scrum no es un proceso recién llegado al panorama internacional, al contrario de lo que ocurre en España donde sí se puede decir que es de reciente adopción. La existencia de Scrum se remonta a principios de la década de los noventa y su origen se sitúa en el desarrollo industrial y en las buenas prácticas adoptadas por un gigante como Toyota. En los últimos años, conforme el desarrollo ágil se ha ido abriendo paso entre los gestores de proyectos de las empresas tecnologías y sus beneficios se han hecho más patentes, Scurm ha sido adoptado por grandes nombres en el concierto internacional como Microsoft, SUN, Google, etc.

Organizar el trabajo

La utilidad fundamental de Scrum es la de organizar el trabajo necesario para la elaboración de un proyecto. Con el objetivo de terminar la construcción del producto a tiempo, satisfaciendo las necesidades del cliente, Scrum propone una serie de reglas, eventos, roles y documentos que permiten satisfacer dicho objetivo. Por ello, es importante dedicarle un apartado a la forma en que el trabajo se gestiona y mide en Scrum.
La primera característica a destacar es cómo gestiona Scrum los requisitos del proyecto. Dichos requisitos, que el producto deberá cumplir, son denominados ítems en la terminología de Scrum. A su vez, cada ítem se descompone en una serie de actividades. El proceso de descomposición se realiza, para cada ítem, durante la reunión previa al sprint en el que se va a acometer su construcción. Los ítems se expresan en lenguaje de negocio mientras que las actividades, que representan una cantidad de trabajo menor, son descritas en lenguaje técnico.
Por su parte, como se ha indicado anteriormente, los ítems son elaborados de forma paulatina, en una serie de sprints. Para conocer cuántos ítems puede el equipo desarrollar durante uno de estos ciclos, se utiliza una medida del trabajo conocida como manday. Esta medida representa el trabajo de una persona a lo largo de una jornada laboral. A modo de ejemplo, si se estima que la elaboración de un ítem puede llevarle a cuatro personas, tres días, se le asignarían doce mandays de duración.
En el siguiente artículo se describirán con más profundidad los pasos que se ejecutan, en cada sprint, para conocer cuántos ítems puede elaborar un equipo, así como los cálculos que, una vez terminado dicho ciclo, se realizan para determinar la productividad alcanzada y hasta qué punto se han cumplido los objetivos planteados.
Por último, un aspecto que es importante destacar en Scrum es la forma en que se produce el seguimiento del desarrollo y de la madurez de las características que se está construyendo. En la mayoría de procesos se miden en función de la cantidad de trabajo que se ha hecho. Por ejemplo, se puede indicar a los desarrolladores que, cada día, apunten las horas que han dedicado a la programación de una determinada característica.
Sin embargo, en Scrum la aproximación es totalmente diferente: el trabajo se gestiona teniendo en cuenta, siempre, la cantidad de él que queda para finalizar una tarea. Esta idea tiene su reflejo en las mediciones de tiempo y en la forma de gestionarlo en los documentos que se utilizan en Scrum, como se verá más adelante.



Tarjeta de Sprint Backlog físico en Word.

Roles y documentos

Los implicados en un proyecto gestionado con Scrum pueden realizar varios roles o papeles distintos. Estos roles van a ser tres: Scrum Master, Product Owner y Equipo. Cada uno de ellos será descrito en profundidad más adelante, indicando tanto sus responsabilidades como los documentos que han de gestionar.
Precisamente en este aspecto, el de la documentación, Scrum hace una nueva demostración de su naturaleza ágil reduciendo el número de artefactos que, por defecto, son necesarios a sólo dos: el Product Backlog, que contiene información sobre todas las características a elaborar para completar el proyecto, y el Sprint Backlog, que es un subconjunto del anterior y contiene los requisitos a desarrollar durante un sprint.
El formato de estos documentos es libre; es decir, no existe una referencia oficial ni de cómo almacenarnos, ni de los campos de información que, por defecto, deben incluir. En lo que respecta a la forma de guardar su información, parece lógico utilizar algún tipo de documento electrónico. De los existentes, las hojas de cálculo suelen ser las preferidas de los gestores de proyecto. Tanto el Product Backlog como el Sprint Backlog tienen una serie de características que los hace candidatos ideales a ser gestionados por alguna aplicación del estilo a Microsoft Excel.
Sin embargo, en Clay Formación Internacional se ha adoptado un enfoque mixto con respecto al Sprint Backlog. Además de hacer uso de Excel para su almacenamiento y gestión, el documento tiene una versión manual o física,



en forma de panel. En dicho panel se colocan una serie de tarjetas que representan a los ítems que se van a elaborar durante el sprint. En la figura 1 se puede ver una imagen de este panel físico, con algunas de las tarjetas colocadas en él. El panel presenta tres columnas, correspondientes a las tareas que están por empezarse (columna izquierda), las que están en proceso (columna central) y las que están terminadas (columna derecha). Los post-its adosados a cada tarjeta corresponden con las actividades de dicha tarea.

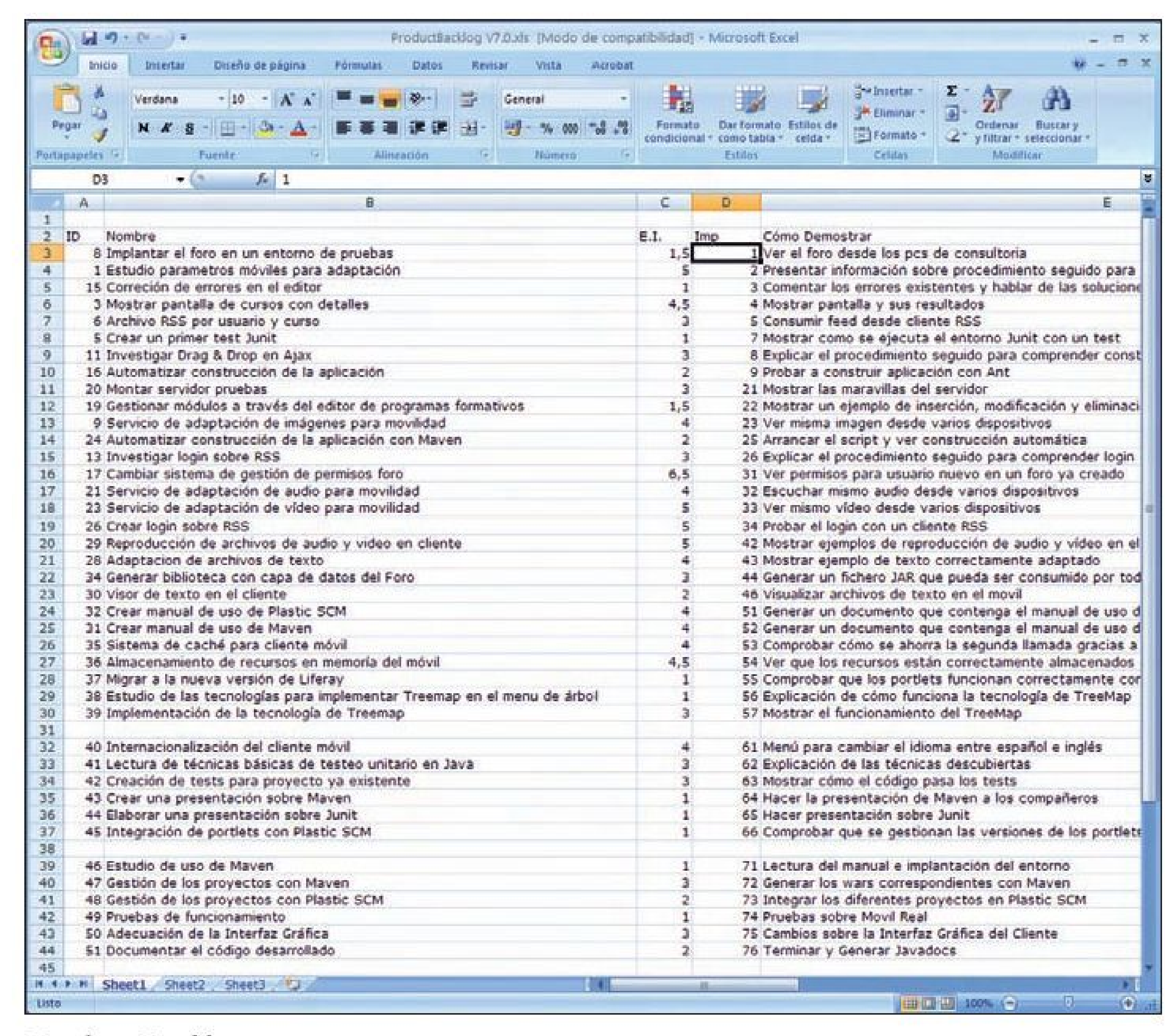
Cada tarjeta contiene una serie de campos que replican la información existente para el ítem en el Product Backlog digital. En Clay se utiliza un proceso de vinculación entre Excel y Word, que da como resultado la creación automática de estas tareas. La combinación de una plantilla Word que contiene una serie de campos especiales, con la información del fichero Excel, da como resultado una serie de tarjetas como las que se pueden ver en la Figura 2, con la información del ítem.

Con esta versión física del Sprint Backlog se consigue que los miembros del equipo interactúen de una forma más eficiente con el documento durante los distintos eventos en que se utiliza, de lo que se logra con una versión digital. Mientras que con un documento electrónico la modificación se reserva a la persona que esté al cargo del teclado, con la versión manual todos los desarrolladores pueden hacer los cambios que estimen oportunos a la vez, sintiéndose a la vez más implicados en el evento por resultar más participativo.

En lo que respecta a los campos de información, hay un cierto acuerdo entre la mayoría de autores relevantes sobre qué campos son los imprescindibles. En el caso del Product Backlog, son los siguientes:

- Id: identificador único para cada ítem.
- Nombre: descripción del ítem, en no más de diez palabras.
- Importancia: número que indica la prioridad en la construcción del ítem. La cantidad no es significativa de la prioridad, sólo una forma de ordenación.
- Estimación inicial: tiempo que lleva el desarrollo del ítem. Se estima antes de introducir el ítem en un sprint y se mide en mandays.
- Cómo demostrar: descripción sobre cómo se ha de presentar el ítem durante la demostración, al final del sprint. Ayuda al desarrollador a la hora de escribir las pruebas de aceptación.

Existen otros muchos campos que se pueden incluir, como serían unas notas para añadir apuntes relacionados con el ítem, un campo con el identificador que tenga el ítem en el sistema de tracking de errores, en caso de que el ítem se trate de un bug, etc. En cualquier caso, es recomendable ser conciso en cuanto a la cantidad de información a utilizar.



ID	Nombre	E.I.	Imp	Cómo Demostrar
8	Implantar el foro en un entorno de pruebas	1,5	1	Ver el foro desde los pcs de consultoria
1	Estudio parametros móviles para adaptación	5	2	Presentar información sobre procedimiento seguido para
15	Correción de errores en el editor	1	3	Comentar los errores existentes y hablar de las solucione
3	Mostrar pantalla de cursos con detalles	4,5	4	Mostrar pantalla y sus resultados
6	Archivo RSS por usuario y curso	3	5	Consumir feed desde cliente RSS
5	Crear un primer test Junit	1	7	Mostrar como se ejecuta el entorno Junit con un test
11	Investigar Drag & Drop en Ajax	3	8	Explicar el procedimiento seguido para comprender const
16	Automatizar construcción de la aplicación	2	9	Probar a construir aplicación con Ant
20	Montar servidor pruebas	3	21	Mostrar las maravillas del servidor
19	Gestionar módulos a través del editor de programas formativos	1,5	22	Mostrar un ejemplo de inserción, modificación y eliminaci
9	Servicio de adaptación de imágenes para movilidad	4	23	Ver misma imagen desde varios dispositivos
24	Automatizar construcción de la aplicación con Maven	2	25	Arrancar el script y ver construcción automática
13	Investigar login sobre RSS	3	26	Explicar el procedimiento seguido para comprender login
17	Cambiar sistema de gestión de permisos foro	6,5	31	Ver permisos para usuario nuevo en un foro ya creado
21	Servicio de adaptación de audio para movilidad	4	32	Escuchar mismo audio desde varios dispositivos
23	Servicio de adaptación de vídeo para movilidad	5	33	Ver mismo vídeo desde varios dispositivos
26	Crear login sobre RSS	5	34	Probar el login con un cliente RSS
29	Reproducción de archivos de audio y video en cliente	5	42	Mostrar ejemplos de reproducción de audio y vídeo en el
28	Adaptacion de archivos de texto	4	43	Mostrar ejemplo de texto correctamente adaptado
34	Generar biblioteca con capa de datos del Foro	3	44	Generar un fichero JAR que pueda ser consumido por tod
30	Visor de texto en el cliente	2	46	Visualizar archivos de texto en el movil
32	Crear manual de uso de Plastic SCM	4	51	Generar un documento que contenga el manual de uso d
31	Crear manual de uso de Maven	4	52	Generar un documento que contenga el manual de uso d
35	Sistema de caché para cliente móvil	4	53	Comprobar cómo se ahorra la segunda llamada gracias a
36	Almacenamiento de recursos en memoria del móvil	4,5	54	Ver que los recursos están correctamente almacenados
37	Migrar a la nueva versión de Liferay	1	55	Comprobar que los portlets funcionan correctamente cor
38	Estudio de las tecnologías para implementar Treemap en el menu de árbol	1	56	Explicación de cómo funciona la tecnología de TreeMap
39	Implementación de la tecnología de Treemap	3	57	Mostrar el funcionamiento del TreeMap
40	Internacionalización del cliente móvil	4	61	Menú para cambiar el idioma entre español e inglés
41	Lectura de técnicas básicas de testeo unitario en Java	3	62	Explicación de las técnicas descubiertas
42	Creación de tests para proyecto ya existente	3	63	Mostrar cómo el código pasa los tests
43	Crear una presentación sobre Maven	1	64	Hacer la presentación de Maven a los compañeros
44	Elaborar una presentación sobre Junit	1	65	Hacer presentación sobre Junit
45	Integración de portlets con Plastic SCM	1	66	Comprobar que se gestionan las versiones de los portlets
46	Estudio de uso de Maven	1	71	Lectura del manual e implantación del entorno
47	Gestión de los proyectos con Maven	3	72	Generar los wars correspondientes con Maven
48	Gestión de los proyectos con Plastic SCM	2	73	Integrar los diferentes proyectos en Plastic SCM
49	Pruebas de funcionamiento	1	74	Pruebas sobre Movil Real
50	Adecuación de la Interfaz Gráfica	3	75	Cambios sobre la Interfaz Gráfica del Cliente
51	Documentar el código desarrollado	2	76	Terminar y Generar Javadocs

Product Backlog.

Los campos del Sprint Backlog también son adaptables a las necesidades de información, aunque se suele coincidir en señalar a los campos id, nombre, importancia y estimación inicial, como suficientes. Junto a estos campos, se adjunta algún tipo de línea de tiempo en la que ir indicando la cantidad de trabajo que resta para finalizar el ítem. Más adelante se verá un ejemplo de esta línea.

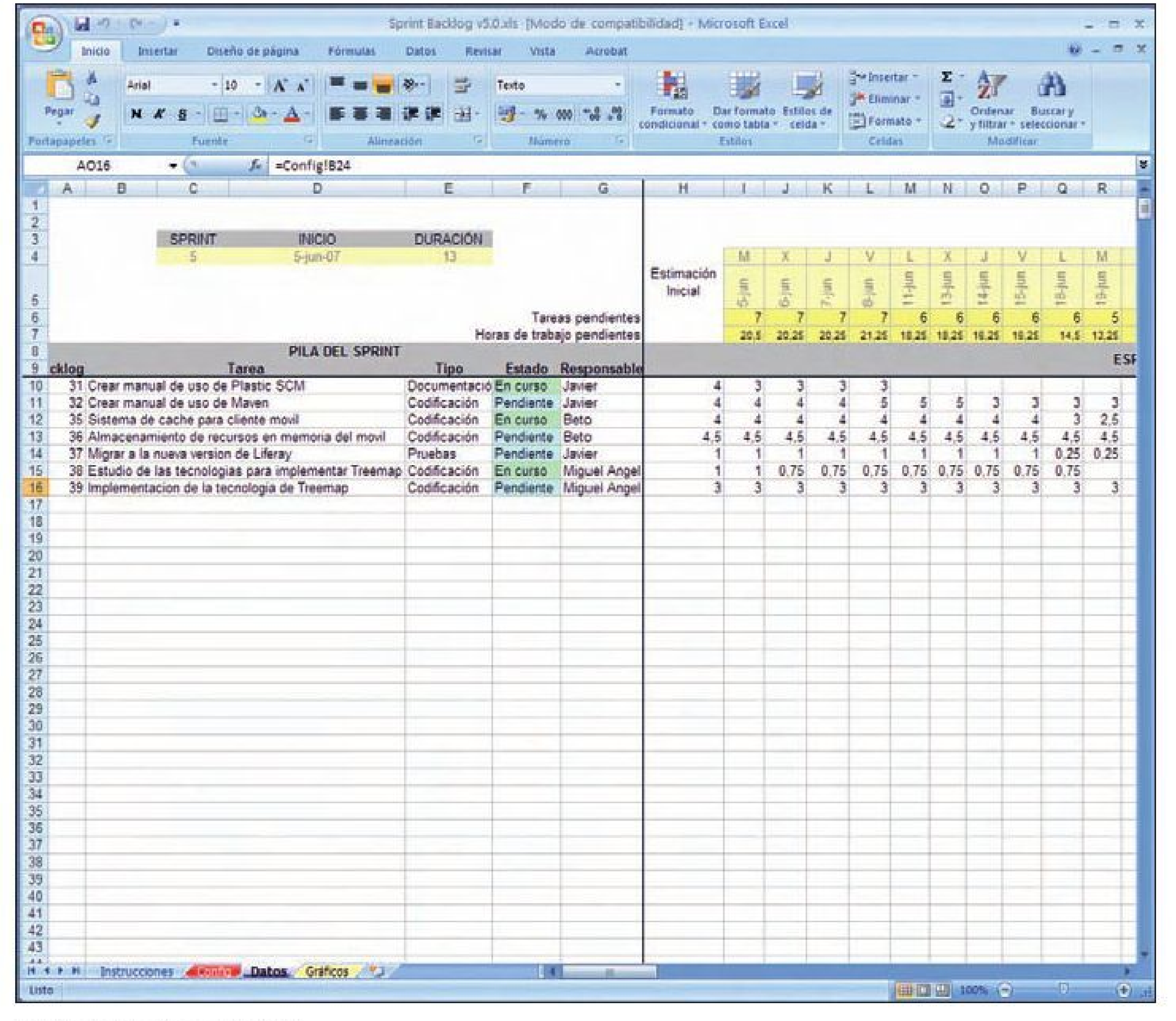
Algunos gestores que adoptan por primera vez Scrum y que vienen de procesos pesados, como por ejemplo RUP (Rational Unified Process), pueden sentir la tentación de llevarse las manos a la cabeza al comprobar hasta qué punto la documentación pasa a ser prescindible cuando se está usando Scrum.

Sin embargo, todo esto no hace sino responder a otro de los pilares de las filosofías ágiles: dar más importancia al software que funciona que a la documentación. Esta idea no puede realizarse con éxito si no es soportándola mediante una serie de prácticas, como son la propiedad colectiva del código, la utilización de estándares de codificación, la refactorización frecuente del código para mejorar su calidad y legibilidad, etc. Todas estas prácticas están propuestas por XP y suelen aplicarse, con bastante frecuencia, en Scrum. La combinación de ambos ofrece muy buenos resultados y suele producirse habitualmente.

En cualquier caso, Scrum siempre deja abierta la puerta a introducir documentación extra que se considere necesaria y relevante para llevar con éxito la gestión del proceso y del trabajo diario. Tal ha sido el caso de la implantación hecha en Clay Formación Internacional, donde se han adoptado dos artefactos extra. El primero de ellos es el documento de revisión del sprint, casi tan popular y generalizado como los dos indicados anteriormente, y que contendrán los resultados de cada sprint.

El otro documento introducido, el Sprint Backlog personal, ha sido una creación ad-hoc que venía a cubrir una necesidad expuesta por los miembros del equipo de Clay. Gracias a este artefacto, les es posible paralelizar su trabajo, algo que en principio no contempla Scrum que, por el contrario, propone la realización de forma secuencial de las tareas de cada miembro, como forma de combatir la pérdida de productividad derivada de los cambios de contexto entre tareas.





SPRINT	INICIO	DURACIÓN
5	5-jun-07	13

cklog	Tarea	Tipo	Estado	Responsable	Estimación Inicial	M 5-jun	X 6-jun	J 7-jun	V 8-jun	L 11-jun	X 13-jun	J 14-jun	V 15-jun	L 18-jun	M 19-jun
				Tareas pendientes		7	7	7	7	6	6	6	6	6	5
				Horas de trabajo pendientes		20,5	20,25	20,25	21,25	18,25	18,25	16,25	16,25	14,5	12,25
31	Crear manual de uso de Plastic SCM	Documentació	En curso	Javier	4	3	3	3	3						
32	Crear manual de uso de Maven	Codificación	Pendiente	Javier	4	4	4	4	5	5	5	3	3	3	3
35	Sistema de cache para cliente movil	Codificación	En curso	Beto	4	4	4	4	4	4	4	4	4	3	2,5
36	Almacenamiento de recursos en memoria del movil	Codificación	Pendiente	Beto	4,5	4,5	4,5	4,5	4,5	4,5	4,5	4,5	4,5	4,5	4,5
37	Migrar a la nueva version de Liferay	Pruebas	Pendiente	Javier	1	1	1	1	1	1	1	1	1	0,25	0,25
38	Estudio de las tecnologias para implementar Treemap	Codificación	En curso	Miguel Angel	1	1	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	
39	Implementacion de la tecnologia de Treemap	Codificación	Pendiente	Miguel Angel	3	3	3	3	3	3	3	3	3	3	3

Sprint Backlog digital.

El Product Owner y el Product Backlog

El Product Owner es uno de los roles novedosos que propone Scrum. La persona encargada de representarlo es aquella que tenga la mejor comunicación con el cliente, aquella que conozca mejor qué necesidades tiene y, por tanto, qué requisitos debe cumplir el producto que se está construyendo. Gracias a este conocimiento, es el responsable de elaborar y gestionar el Product Backlog, el documento que contiene los ítems que se construirán a lo largo del proyecto.

El rol de Product Owner debe ser siempre ejercido por un único responsable. De esta manera, en cada equipo habrá una única persona que pueda decirle a los demás qué características tiene que tener el producto. Es muy común el escenario en el que varias personas hacen sugerencias o indicaciones a los desarrolladores sobre qué ha de incluirse en el producto. La tendencia habitual a que personal de ventas, de instancias superiores o, en general, de cualquiera que tenga algún tipo de interés en el resultado final del proyecto, se permitan imponer características al equipo, es totalmente eliminada en Scrum.

Cada nuevo requisito, sugerencia o comentario debe pasar el filtro del Product Owner, puesto que sólo a través de él puede entrar cualquiera de ellos en la espiral de construcción del producto. Esta situación supone un gran alivio para los desarrolladores, que ven desaparecer una buena cantidad de injerencias que merman su productividad y les impide mantenerse focalizados en construir los ítems correspondientes al sprint.

En aquellas ocasiones en las que el Product Owner se vea altamente presionado para incluir una nueva característica en el ciclo actual, puede recurrir a la detención del sprint. En esta situación, se vuelven a dar los pasos habituales para crear un sprint, incluyendo en esta ocasión el nuevo ítem en el Sprint Backlog. Sin embargo, este proceso produce una gran pérdida de tiempo, por toda la sobrecarga relacionada con el inicio de un nuevo sprint. El retraso suele ser suficiente motivo para que aquellos que estén presionando para incluir el nuevo requisito, se planteen esperar al siguiente ciclo.

El Product Owner realiza su labor apoyado en el Product Backlog. Este artefacto es exclusiva responsabilidad suya, siendo el único que puede modificarlo. Esta idea refuerza el hecho de que nadie, excepto el Product Owner, puede introducir nuevos requisitos en el proyecto, puesto que cada uno de ellos debe tener reflejo en el Product Backlog. En la figura 3 se puede ver una imagen de un Product Backlog realizado con Excel.

Aunque sólo el Product Owner pueda modificar el Product Backlog, la visibilidad del mismo es para todo el equipo e, inclusive, para cualquier miembro de la organzación interesado en el proyecto. Siendo como es la lista de requisitos del producto, este documento realiza una labor informativa sobre qué se está construyendo y qué características va a tener. La comprensión de los ítems tampoco será un problema para aquellos que utilicen el Product Backlog como fuente de información, puesto que, como se indicó anteriormente, los ítems están expresados en el lenguaje de negocio propio de la aplicación.

Esto último lleva a una característica peculiar del Product Owner: no es imprescindible que sea una persona con conocimientos técnicos, puesto que su comunicación, tanto con el cliente como con el equipo, la va a realizar siempre en el lenguaje de negocio. En cualquier caso, la tendencia actual es que los Product Owner sí tengan conocimientos técnicos, puesto que así pueden lograr una comunicación más fluida con el equipo y entender en su totalidad el trabajo que se está realizando.

El Scrum Master y el Sprint Backlog

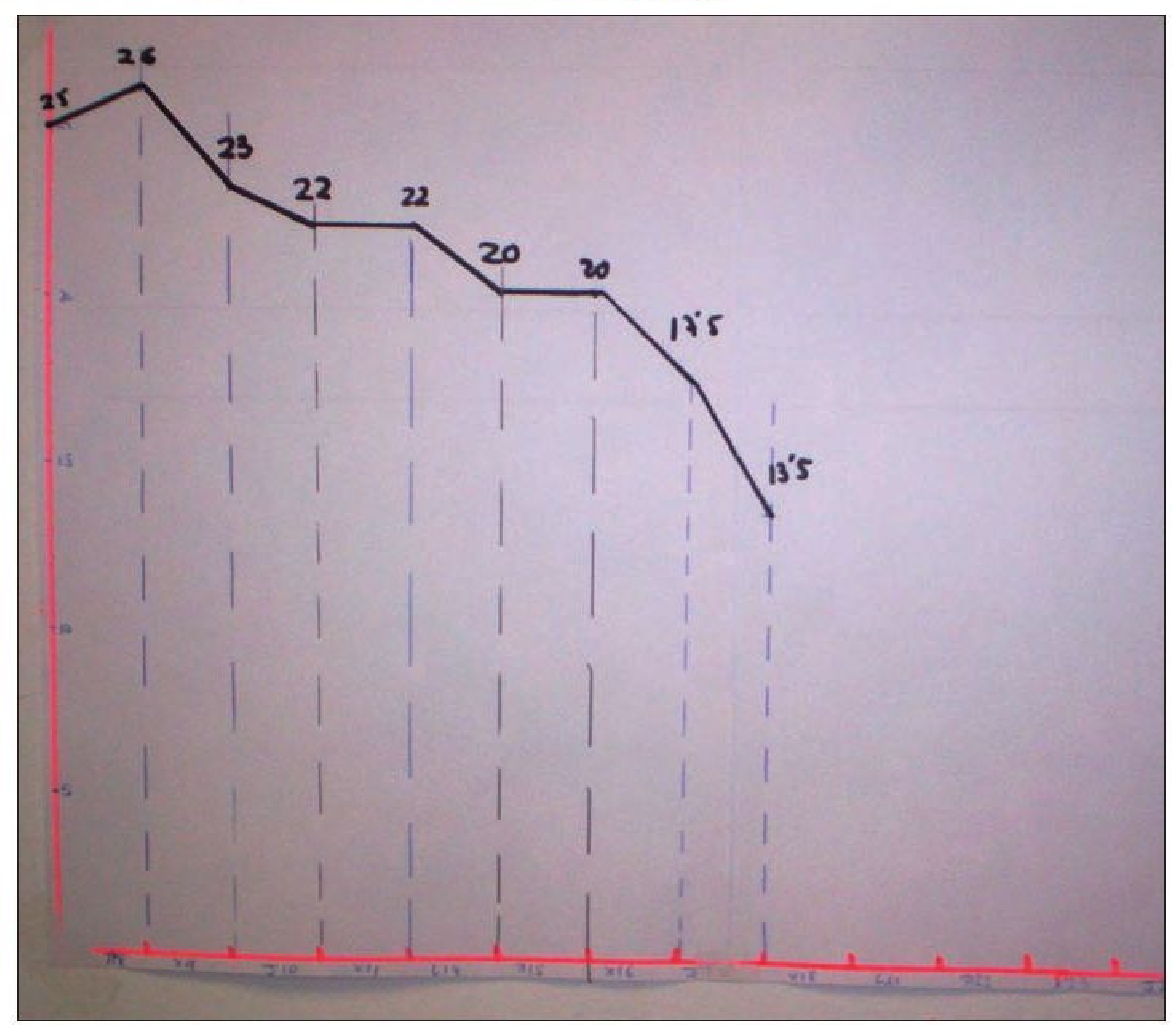
El Scrum Master es otro de los innovadores roles que propone Scrum. Su figura se podría asemejar a la del jefe de proyecto de los procesos clásicos, aunque sus responsabilidades no son del todo equivalentes.

Entre las labores que conserva, con respecto a las que ejerce un jefe de proyecto tradicional, está la organización del trabajo de que se compone la elaboración del proyecto. Sin embargo, la forma de ejercer esta tarea es distinta a la habitual. En el caso de Scrum, lleva a cabo dicha labor en colaboración con el Product Owner. Juntos trabajan en la creación del Sprint Backlog de cada sprint, el documento con el que el Scrum Master gestiona la evolución del trabajo durante el siguiente ciclo.

Otra responsabilidad del Scrum Master, en este caso que le diferencia del jefe de proyecto habitual, es la de dar soporte al equipo y ayudarle a autoorganizarse. El soporte lo provee, fundamentalmente, asegurándose de resolver cualquier impedimento que dificulte el trabajo diario del equipo y merme su productividad. En cuanto a la autoorganización, debe poner todo lo que esté de su parte para que las propuestas y medidas que adopte el equipo sean respetadas por todos los esta-



mentos de la empresa, reforzando así dicha capacidad de toma de decisiones.
También es cometido del Scrum Master controlar los distintos eventos que se suceden en Scrum, especialmente en lo que respecta a la duración de los mismos y el cumplimiento de los objetivos que, en cada uno de ellos, se plantean. En este sentido, el Scrum Master debe asegurarse de que el proceso es seguido con rectitud por sus participantes, puesto que a pesar de ser Scrum un framework altamente adaptable, una vez que sus reglas han sido ajustadas deben seguirse al pie de la letra.
Por último, son tareas del Scrum Master todas aquellas relacionadas con la creación y posterior gestión del Sprint Backlog, en cuya confección también participa el Product Owner como se ha indicado antes. Este documento va a contener un conjunto de ítems que han de ser desarrollados durante las semanas que dure el sprint. En la reunión inicial previa a cada sprint (y de la que se hablará en profundidad en el siguiente artículo), el Scrum Master y el Product Owner seleccionan un conjunto de funcionalidad a desarrollar en el siguiente sprint, de entre las características incompletas que se encuentren en el Product Backlog.
Tomando como referencia la prioridad que tienen dichos ítems en el Product Backlog, se realizan las estimaciones del tiempo necesario para terminar su desarrollo y se completan los campos de información que falten por rellenar. En cualquier caso, estas estimaciones deben ser actualizadas y confirmadas por los miembros del equipo, en una reunión posterior en la que también van a estar presentes el Product Owner y Scrum Master. El equipo tiene la última palabra con respecto al tiempo necesario para elaborar una característica.
Mediante el Sprint Backlog, el Scrum Master puede realizar un seguimiento del trabajo que falta para cada ítem que ha de desarrollarse durante el sprint. Tras cada reunión diaria, actualizará las cantidades de trabajo en función de las indicaciones que le hagan los miembros del equipo. Por cada nueva actividad de las que componen la tarea finalizada, restará en la correspondiente línea de tiempo asignada a cada una de ellas la cantidad de trabajo que suponía dicha actividad. La figura 4 muestra un ejemplo de realización del Sprint Backlog, pudiéndose observar en la parte derecha de la imagen las distintas líneas de tiempo para cada ítem y cómo la cantidad de trabajo va decreciendo, según avanzan los días en el sprint.



Sprint Backlog Graph.

El Sprint Backlog es responsabilidad única del Scrum Master, siendo el único que puede modificarlo. Es, por tanto, el único con la potestad de añadir o eliminar ítems del documento, aunque debe consensuar esta clase de decisiones con el Product Owner. En cualquier caso, si durante un sprint, el desarrollo de un ítem lleva al descubrimiento de nuevas características que también han de ser construidas, es preferible incluirlas como ítems en el Product Backlog, a la espera de entrar en posteriores sprints.
Junto con la actualización del Sprint Backlog, el Scrum Master debe poner al día también el Sprint Backlog Graph, para mantener ambos sincronizados. Este último es un gráfico descendente que representa, de forma muy visual, cómo va disminuyendo el trabajo que falta por completar, según avanza el sprint. La figura 5 presenta un ejemplo de este gráfico.
Este gráfico tiene una segunda utilidad, especialmente apreciada para aquellos Scrum Master que ya cuenten con una amplia experiencia a sus espaldas. Mediante el análisis de los sucesivos gráficos que resultan tras varios sprints, el Scrum Master puede sacar pautas de comportamiento del equipo. Por ejemplo, si comprueba que la cantidad de trabajo se reduce drásticamente al final de cada ciclo, puede concluir que los desarrolladores se apresuran al llegar al final del sprint para completar sus tareas, lo que probablemente redunde en una falta de calidad de las mismas por la precipitación.

El Equipo y el Sprint Backlog personal

El rol del Equipo es ejercido por todos aquellos desarrolladores comprometidos con el proyecto, que no desempeñen ninguno de los dos roles anteriores. Para ilustrar lo que significa compromiso, Ken Schwaber relata en una de sus obras una pequeña historia, que habla sobre gallinas y cerdos. Una gallina y un cerdo proponen crear un restaurante, pero a la hora de ponerse a ello, la gallina muestra interés, aportando sus huevos, mientras que el cerdo muestra verdadero compromiso, aportando su propia carne en forma de bacon.
En este caso, el paralelismo con un proyecto de software, es claro. Las gallinas estarían representadas por personal de marketing, ventas, alta dirección, etc. Gente que está interesada en el proyecto y en que llegue a buen puerto. En cambio, los desarrolladores, aquellos que pelean diariamente para sacar el trabajo adelante, estarían representados por los cerdos, puesto que su nivel de implicación es mucho mayor.



El compromiso define los límites del equipo en Scrum. Es decir, todos aquellos participantes en el proyecto que estén realmente comprometidos serán miembros del equipo. Puesto que este nivel de compromiso demandado por Scrum suele encontrarse sólo en los desarrolladores, son estos los que realizan este rol.

El equipo debe ser autosuficiente y tomar decisiones por sí mismo, como ya se indicó en la descripción del Scrum Master. Estas decisiones han de ser respetadas por todos los estamentos relacionados con el proyecto, puesto que nadie mejor que aquellos que construyen el software para saber cómo debe hacerse y para estar comprometidos con que el resultado sea lo mejor posible.

Sólo mediante un respeto escrupuloso a estas decisiones, se logra que el equipo alcance esta autosuficiencia y, como consecuencia de ella, su máximo nivel de productividad. Para mantener este grado, como ya se indicó antes, el Scrum Master está obligado a eliminar los impedimentos que puedan surgir en el trabajo diario y el Product Owner se ha de asegurar de filtrar toda clase de injerencias externas. La combinación de ambos esfuerzos, junto con la confianza ganada por el equipo sintiéndose sus decisiones respetadas, ofrece los resultados más óptimos.

En cuanto a la documentación que utiliza el equipo, en principio no existe ningún documento asignado por defecto ni a los miembros de forma individual, ni al equipo en su conjunto. Sin embargo, las posibilidades de personalización que ofrece Scrum llevaron a crear, en la implantación hecha para Clay Formación Internacional, de un documento nuevo: el Sprint Backlog personal. Cada uno de los miembros del equipo tiene la responsabilidad de crear y gestionar este documento, uno para cada sprint en el que participa.

Este documento tiene una estructura muy similar a la del Sprint Backlog que maneja el Scrum Master. Sin embargo, a diferencia del anterior, sólo contiene los ítems asignados al desarrollador y, para cada ítem, indica también las actividades de que se compone. En la figura 6 se puede ver un ejemplo de Sprint Backlog personal. Las líneas resaltadas en negrita corresponden a las tareas, mientras que las líneas que se sitúan debajo son las actividades que conforman cada tarea.

Gracias a esta estructura, al desarrollador le resulta muy sencillo trabajar en dos actividades de dos ítems distintos en un mismo día y asignar, para cada una, la cantidad de tiempo que le ha dedicado. Este aspecto no está contemplado en Scrum, donde se propone realizar el trabajo de forma secuencial, algo con lo que muchos desarrolladores no se sienten a gusto.

SPRINT	INICIO	DURACIÓN
5	5-jun-07	13

PILA DEL SPRINT

ckklog	Tarea	Tipo	Estado	Responsable	Estimación Inicial	M 5-jun	X 6-jun	J 7-jun	V 8-jun	L 11-jun	X 13-jun	J 14-jun	V 15-jun	L 18-jun
				Tareas pendientes								1	1	1
				Horas de trabajo pendientes								0,3	0,1	0,2
37	**Migrar a la nueva versión de Liferay**	Pruebas	En curso	Miguel Ángel	1									
	Adaptar scripts de Jorge		En curso		0,75									
	Pruebas		Pendiente		0,25									
38	**Estado de la tecnología para implementar Treemap en**	Investigación	Pendiente	Miguel Ángel	1									
	Evaluar las tecnologías disponibles		Terminada		0,25		0.05	0.20						
	Miniejemplo de prueba		Terminada		0.5									
	Documentacion		Terminada		0,25									
39	**Implementación de la tecnología de treemap**	Codificación	En curso	Miguel Ángel	3									
	Adecuación de los datos		Terminada		1							0,3	0,1	0,2
	Implementación del soporte visual		Terminada		1									
	Variación de la salida		Terminada		0,5									
	Pruebas		Pendiente		0,25									
	Documentación		Pendiente		0,25									

Sprint Backlog personal.

La forma en que se estima el tiempo en el Sprint Backlog personal es otra de las diferencias con respecto al Sprint Backlog y, en general, a la filosofía que se sigue en Scrum. Se indicó anteriormente que en Scrum el trabajo se gestiona con respecto al que resta por hacer para terminar una tarea. Esta regla de oro se rompe en el Sprint Backlog personal, puesto que los desarrolladores apuntan, para cada día, cuántas horas han dedicado a cada actividad. La razón de este cambio la encontramos en la facilidad que aporta a la hora de, cuando han terminado una actividad, poder conocer con exactitud si les llevó el tiempo que habían estimado inicialmente o, por el contrario, erraron en sus predicciones.

Conclusiones

En este primer artículo se ha visto una panorámica general de Scrum, a excepción de los eventos que se realizan a lo largo del proyecto, que se dejan para el siguiente artículo.

Scrum es un framework ágil que, gracias a la adaptación de sus reglas, permite obtener un proceso personalizado para cada ambiente empresarial. Introduce varias novedades con respecto a los procesos clásicos, tanto en la forma de trabajo como en la filosofía del desarrollo.

Cuenta con el respaldo de la industria, puesto que algunos de los pesos pesados del desarrollo de software a nivel mundial, lo han adoptado en sus proyectos. Sin embargo, su alta capacidad de adaptación le permite encajar perfectamente en organizaciones de todo tipo, desde gigantes como Google a pequeñas y medianas empresas como Clay Formación Internacional, en cuyo departamento de I+D+i se ha introducido con éxito.

Una vez vistos los roles que se ejercen en Scrum, junto con los documentos que manejan, el lector tiene información suficiente para, en el siguiente artículo, ver cómo todas estas piezas encajan. En este segundo artículo se describirán los distintos eventos que se celebran en Scrum, así como qué uso se hace de los artefactos ya descritos. Se indicarán también, para cada evento, qué participantes tiene y qué responsabilidades ejerce cada uno. SP

Referencias

[1] Agile Software Development With Scrum, Ken Schwaber and Mike Beedle, Editorial Prentice Hall.

[2] Agile Project Management With Scrum, Ken Schwaber, Editorial Microsoft Professional.

[3] Scrum And XP From The Trenches, Henrik Kniberg, Editorial InfoQ.

SUSCRIBETE A SOLO PROGRAMADORES



Suscripción a Sólo Programadores

SUSCRIPCIÓN PARA ESPAÑA

- Opción A: Suscripción anual con 25% descuento:
  54 euros* (3 revistas gratis, 18 euros de ahorro)
- Opción B: Suscripción anual con 15% descuento:
  61,20 euros* (tapas de regalo, 10,80 euros de ahorro)

*10 euros de gastos de envío para la opción contrareembolso

SUSCRIPCIÓN PARA EXTRANJERO

- Opción C: Suscripción anual con 25% descuento
  Europa: 78 euros (gastos de envío incluidos)
- Opción D: Suscripción anual con 25% descuento
  Resto de paises: 102 euros (gastos de envío incluidos)

FORMAS DE PAGO PARA ESPAÑA

- Contrareembolso *(10 euros gastos de envío)*
- Giro Postal a Revistas Profesionales, S.L.
- Transferencia al Banco Popular:
  c.c:0075/1040/43/0600047439
- Talón Bancario a nombre de Revistas Profesionales
- Domiciliación Bancaria
- Tarjeta de Crédito

FORMA DE PAGO EXTRANJERO:

- Tarjeta de crédito

Suscríbase en www.revistasprofesionales.com Más información en el teléfono 91 304 87 64, en el fax 91 327 13 07 y en rpsuscripciones@revistasprofesionales.com



Programación con Múltiples Hilos (Threads) en Visual Basic; C# y Java (I)

MG. LIC. GASTÓN C. HILLAR - gaston_hillar@hasa.com.ar

¿Cómo aprovechar al máximo los dos, cuatro o más núcleos de ejecución disponibles en los microprocesadores modernos en Visual Basic; C# y Java, sin morir en el intento? Este pregunta es la que vamos a responder en una serie de entregas, destinadas a seres humanos mortales que deseen llevar al máximo a los microprocesadores con múltiples núcleos de ejecución contemporáneos y a los que están por venir.

Sin embargo, no nos detendremos únicamente en los conceptos teóricos, sino que presentaremos ejemplos concretos en los tres lenguajes de programación más populares y de esta manera, veremos cómo es posible optimizar los algoritmos para que se dispersen en los hilos *(threads)* de ejecución adecuados para cada micro-arquitectura y así lograr una muy buena eficiencia sin que el esfuerzo de desarrollo sea inaceptable.

El Paralelismo llegó para quedarse

El multiprocesamiento no es algo nuevo, está presente desde hace muchos años, pero, casi siempre limitado a potentes servidores y a estaciones de trabajo muy específicas, debido al alto costo de los componentes y de los equipos que soportaban más de un microprocesador.

Tuve la suerte de tener mis primeras experiencias con el multiprocesamiento con un equipo de Sun Microsystems, de la línea Enterprise, equipado nada menos que con 32 microprocesadores, hace unos 8 años, es decir, durante el año 2000. No voy a negar que he pasado muchas noches enteras haciendo cuanta prueba de rendimiento pudiera ante semejante monstruo, pues hacía mucho tiempo que investigaba sobre arquitecturas de hardware, pero nunca había podido tener esa clase de equipos disponibles para investigarlo tan a fondo. Allí descubrí lo importante que es para un desarrollador tener buenos conocimiento de las arquitecturas de hardware para los cuales está desarrollando, por más que se trabaje en lenguajes de programación de alto nivel.

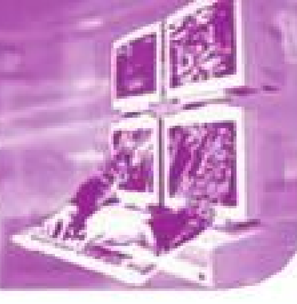
Pues, ese equipo con 32 microprocesadores se comportaba prácticamente exactamente igual que el Pentium III que tenía en mi escritorio en ese momento si se ejecutaba un proceso que utilizaba solamente un hilo de ejecución *(thread)*. Pues, en ese caso, solamente utilizaba uno de sus 32 microprocesadores y los 31 restantes se mataban de risa, totalmente disponibles.

No importaba si se utilizaba C; C++; Java o cualquier otro lenguaje de programación que soportara el sistema operativo Solaris que utilizaba este equipo. Si no se separaba un proceso en múltiples hilos, solamente un microprocesador iba a funcionar y el planificador iba a ir pasando a ese hilo de un microprocesador a otro en algunas ocasiones, sin que ello generara ningún tipo de mejora en el tiempo final del proceso.

Ahora bien, el problema surgía debido a que era necesario que un proceso que tardaba unos 5 días y medio enteros ejecutándose (132 horas aproximadamente) en un solo microprocesador se completara en no más de 8 horas. Por más optimización que se le hiciera al código, aunque se programara directamente en lenguaje ensamblador *(assembler)*, era técnicamente imposible conseguir reducir en casi 30 veces su tiempo de ejecución. La única manera era organizar ese proceso para que generara tantos hilos de ejecución *(threads)* como procesadores encontrara disponibles y así procesar en forma simultánea cada uno de los hilos. Claro que esto requirió trabajar con múltiples hilos de ejecución y todo lo que ellos implica, nada menos que en el lenguaje de programación C++. El resultado fue el esperado, el tiempo total del proceso fue de poco más de 5 horas.

Si el código no se optimizaba para adecuarlo a la potencia de paralelismo que presentaba la arquitectura de hardware, el proceso tardaba las 132 horas mencionadas anteriormente.

La tendencia actual de los microprocesadores de agregar más núcleos de ejecución y de no aumentar tanto la frecuencia de trabajo, más allá de que van mejoran-



do la relación de instrucciones por cada ciclo, apuntan claramente a seguir mejorando el paralelismo. Al menos, salvo que se produzca algún descubrimiento completamente revolucionario, en los próximos años veremos cómo los microprocesadores multiplican sus núcleos de ejecución, pero no crecerán exponencialmente en sus frecuencias de trabajo. Por lo cual, llegó la hora que los desarrolladores acoplen mucho mejor las aplicaciones a las nuevas arquitecturas de paralelismo presentes en los ordenadores modernos. Para poder conseguirlo, en estas entregas vamos a ir viendo ejemplos concretos de las situaciones más comunes en las cuales necesitaremos optimizar para aprovechar los núcleos de procesamiento disponibles y como lo veremos para Visual Basic .Net 1; 2005 y 2008; C# .Net; 2005 y 2008 y para Java 5 y 6, estaremos cubriendo los lenguajes de programación más utilizados en los tiempos que corren, al menos para el desarrollo de aplicaciones de escritorio o de aquéllas que correrán sobre servidores. Por supuesto, que los mismos conceptos que veremos en estos lenguajes de programación, se aplicarán a cualquier otro lenguaje que soporte los múltiples hilos de ejecución.

Estamos sin lugar a dudas en un punto de inflexión en el cual, si los desarrolladores no aprendemos a aprovechar la potencia del procesamiento en paralelo, perderemos un tren en el cual difícilmente luego nos podamos volver a subir. Hace recordar a los tiempos en los cuales muchos desarrolladores que trabajaban muy bien con bases de datos con el sistema operativo DOS, no pudieron pasarse más al mundo de los entornos gráficos (Windows, Linux, etc.).

Estamos frente a una situación crítica en la cual, la gran mayoría de las aplicaciones que se ejecutan en los ordenadores modernos con múltiples núcleos de ejecución, cuando se ven exigidas al 100%, hacen trabajar únicamente a un núcleo de procesamiento, independientemente de la cantidad que tenga disponible el equipo. Se está desaprovechando una gran cantidad de recursos de procesamiento por no tener la programación adecuada. Y esto no se limita al software casero, sino que podemos extenderlo a los productos más elaborados.

Vamos a tomar por ejemplo un microprocesador Intel Core 2 Quad Q6700, el cual posee cuatro núcleos de ejecución. En la mayoría de los casos, si utilizamos la herramienta de monitoreo de los recursos del procesador del sistema operativo en el cual estemos trabajando cuando una aplicación está demorando un tiempo en darnos la respuesta, es decir, cuando se la está exigiendo al máximo, obtendremos una imagen

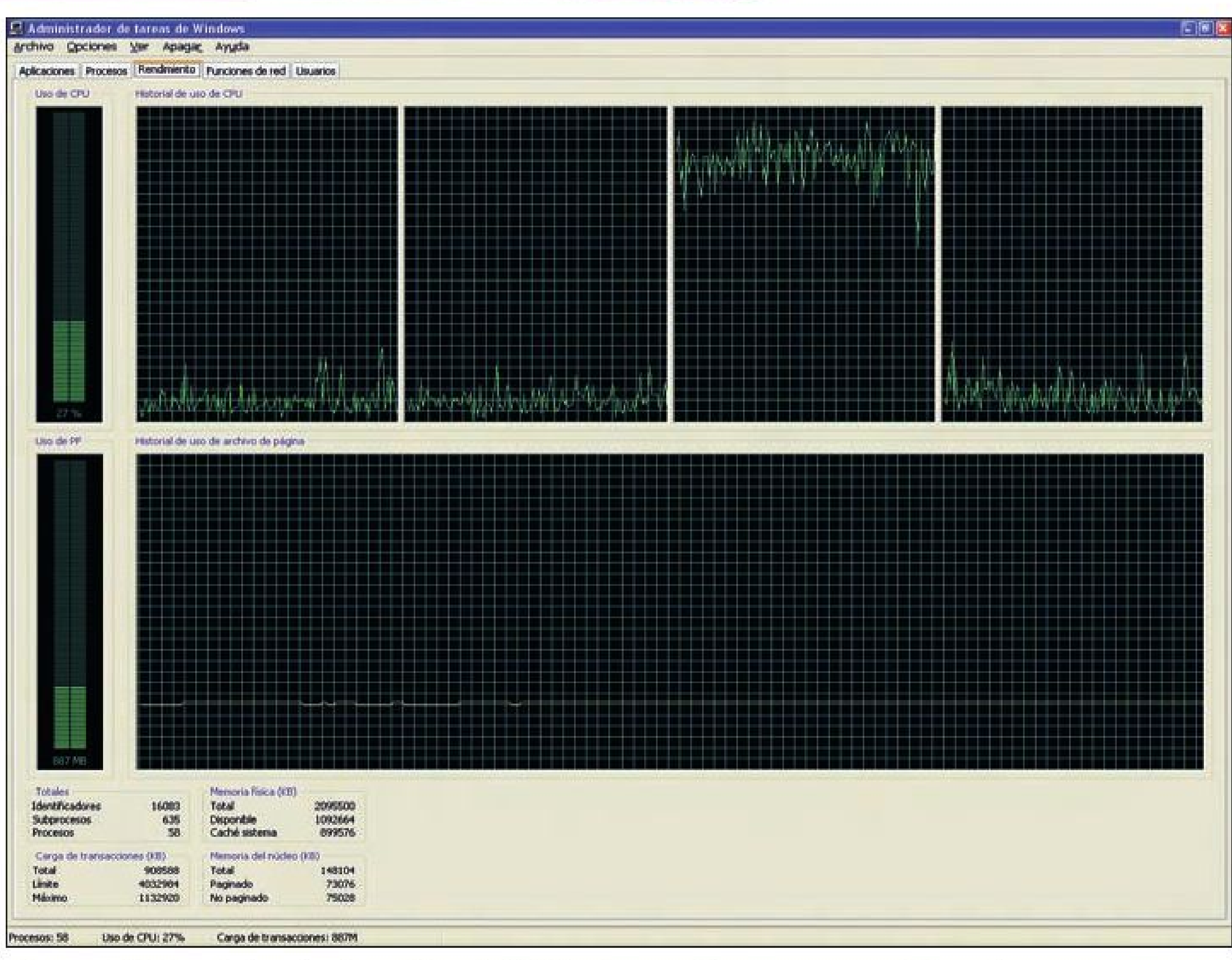


Figura 1. Un núcleo de ejecución al 100% mientras los otros tres núcleos descansan en paz.

similar a la Figura 1. El objetivo de estas entregas es conseguir aplicaciones que lleven al máximo a todos los núcleos deseados, dependiendo de la configuración de hardware frente a la cual estamos y a las preferencias del usuario de la misma. Además, se presentarán patrones de diseño orientados a las situaciones que mejor se pueden resolver con múltiples hilos y tips y recomendaciones para el multiprocesamiento futuro, pues el paralelismo llegó para quedarse.

El cambio de paradigma en la arquitectura de software

Toda revolución genera ciertamente un cambio de paradigma para poder enfrentarla. La masificación de los microprocesadores con múltiples núcleos y la necesidad de dividir los algoritmos más complejos en diferentes hilos de ejecución llevan también a esta necesidad que cada cierta cantidad de años se presenta inexorablemente ante la comunidad de desarrolladores.

La mejor forma de comprender las diferencias entre el monoprocesamiento (un único procesador con un solo núcleo de ejecución) y el multiprocesamiento (varios núcleos de ejecución en uno o más microprocesadores físicos) es viendo cómo ciertos mecanismos de aprovechamiento de la capacidad de procesamiento de un modelo se desmoronan al utilizarlo en el multiprocesamiento y luego, aplicando los cambios de paradigma necesarios, reorganizar la arquitectura de una solución para que pueda sacar el máximo jugo de todos los núcleos disponibles.

Estos ejemplos en los cuales vamos a trabajar son totalmente independientes de la plataforma en la que se ejecuten. Por ejemplo, si tomamos los ejemplos en el lenguaje de programación Java, no importa si los ejecutamos sobre Solaris; GNU Linux; FreeBSD; Mac OS X; Windows XP; Windows 2003; Windows 2008 o Windows Vista, entre otros. Por más que un núcleo y su planificador de procesos sea más o menos eficiente en mayor o menor grado, el aprovechamiento de los núcleos de ejecución para una aplicación contenida en un único proceso será totalmente dependiente de la arquitectura de la solución y del manejo de hilos de ejecución que hagamos en ella. Por el momento, ningún planificador de un núcleo de un sistema operativo existente puede dividir el procesamiento en paralelo en diferentes núcleos en forma automática, totalmente independiente de cómo esté el código máquina del proceso en ejecución. Es decir, es total responsabilidad de los desarrolladores.

Claro que, por supuesto, pueden existir diferencias entre cada una de las plataformas acorde a la disponibilidad de recursos de hardware y a su uso eficiente, acorde al contexto en el cual se esté ejecutando la aplicación. Sin embargo, los principios que analizaremos se aplican por igual a todas ellas. Si bien desde el mismo nacimiento del multiprocesamiento existen grandes esfuerzos por librar a los desarrolladores de las complejidades de la planificación en paralelo de la ejecución de las tareas, no ha habido logros extraordinarios que hagan que no valga la pena estudiar el tema en detalle. Es más, en la actua-



lidad, uno de los grandes frenos en el crecimiento del rendimiento de las aplicaciones a nivel mundial es la falta de desarrolladores capacitados en sacar provecho de las nuevas arquitecturas de procesamiento paralelo disponible. Por lo cual, valdrá mucho la pena tomarnos el tiempo de aprender las técnicas necesarias para diferenciarnos, pues redundará en grandes oportunidades profesionales por un lado y también, por qué no, en el logro de retos personales de poder conseguir hacer que muchos procesos se ejecuten en menos tiempo.

Vamos a tomar un ejemplo clásico de la forma de llevar a cabo un proceso determinado a muchos elementos sin implementar un paralelismo y veremos cuán ineficiente es al estar frente al multiprocesamiento en cualquiera de sus formas. En el Listado 1 presentamos un bloque de código clásico de un ciclo que se ejecutará 36 millones de veces, generando en la variable s del tipo String, un carácter del código Unicode entre 0 y 255. Este tipo de lazos, ejecutados en un equipo con monoprocesamiento (con un único núcleo de ejecución), hacen que la herramienta de análisis del rendimiento del microprocesador del sistema operativo en el cual lo ejecutemos muestre un uso casi ininterrumpido del 100% del mismo, hasta que el lazo termine. Los procesos repetitivos son los que suelen hacernos esperar, pues se deben llevar a cabo una gran cantidad de tareas hasta que volvemos a tener el control de la aplicación.

Ahora bien, si compilamos ese mismo código y lo ejecutamos en un microprocesador con 2 núcleos de ejecución (doble núcleo), como un Intel Core 2 Duo E6750 o un AMD Athlon 64 X2 6000+, veremos como el uso neto del microprocesador no supera el 50% o el 55% si el sistema operativo está haciendo alguna tarea de fondo. Si en la herramienta de monitoreo del microprocesador visualizamos un gráfico por cada núcleo o CPU (UCP), el planificador de procesos puede distorsionar los datos y confundirnos. Pues, tomando como base el ejemplo anterior, si visualizamos un gráfico por cada núcleo, es posible que veamos en el monitoreo que cada uno de los dos núcleos van teniendo una carga equivalente, pero ninguno de los dos llega casi al 100% en forma simultánea. Es decir, parecería como si se fueran turnando el proceso. Es por ello que, para no distorsionar las estadísticas, lo mejor en estos casos es visualizar el uso neto del microprocesador. Si tenemos uno sostenido de entre el 50 y el 55% (ver la Figura 2) y tenemos dos núcleos, sin lugar a dudas, hay uno de ellos que está prácticamente con el 95% de su potencia de procesamiento disponible, sin utilizar,

LISTADO 1 Códigos de ejemplo de un lazo clásico del monoprocesamiento

```
Lenguaje de Programación Visual Basic:
Dim i As Long
Dim s As String

For i = 1 To 36000000
    ProgressBar1.Value = ((i / 36000000) * 100)
    s = Chr(i Mod 255)
    Application.DoEvents()
Next
Lenguaje de Programación C#:
long i;
String s;
char miChar;

for (i=1; i<=36000000; i++) {
    miChar = (char) (i % 255);
    s = miChar.ToString();
    progressBar1.Value = (int)((i / 36000000) * 100);
    Application.DoEvents();
}
Lenguaje de Programación Java:
long i;
String s;
char miChar;

for (i=1; i<=36000000; i++) {
    miChar = (char) (i % 255);
    s = Character.toString(miChar);
    jProgressBar1.setValue((int)((i / 36000000) * 100));
}
```

pues la aplicación no lo aprovecha para nada. La actividad que refleja se debe a tareas que está realizando el sistema operativo u otras aplicaciones que corren en segundo plano.

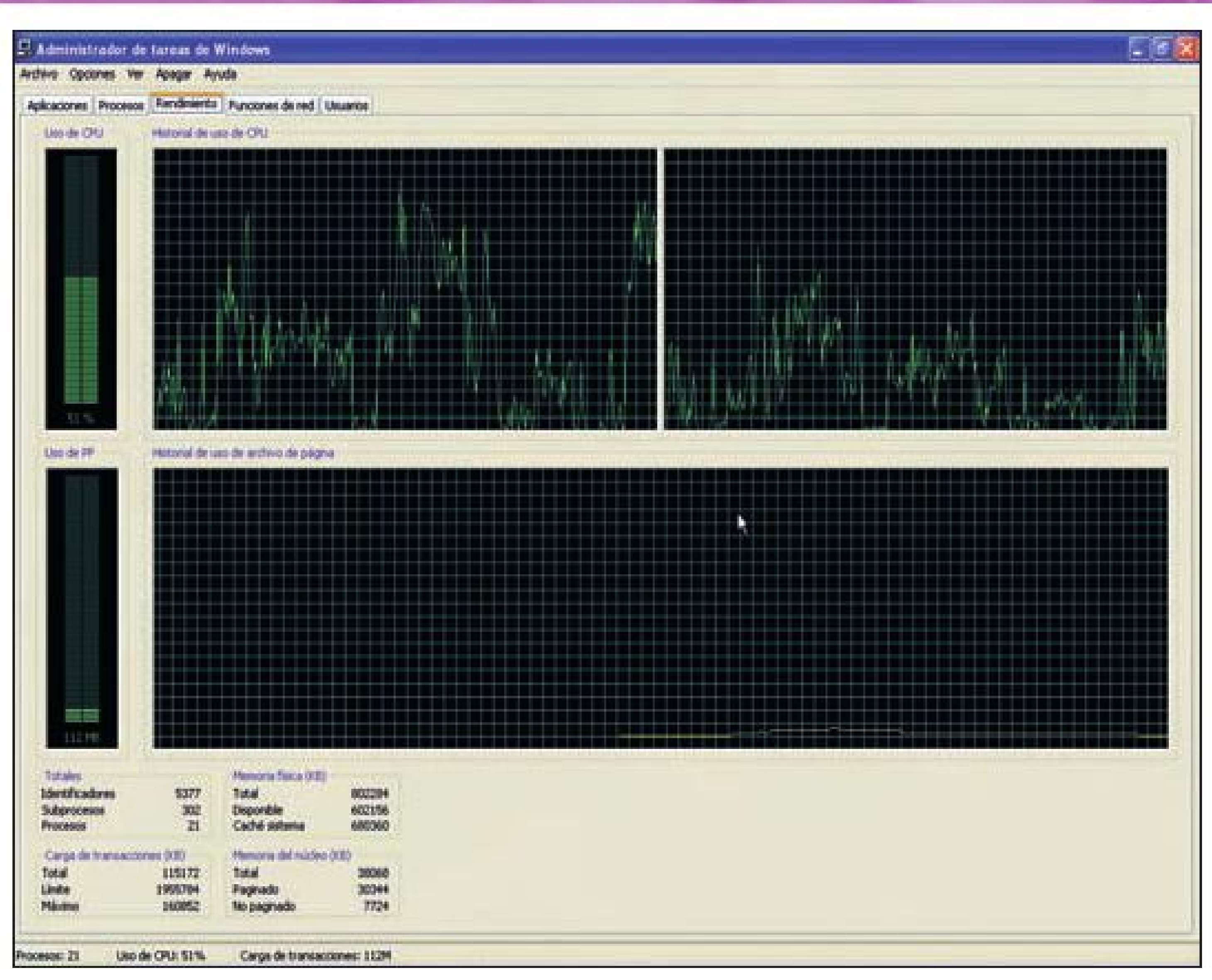
Ahora, conociendo los resultados del caso anterior, qué pasaría si, compilamos ese mismo código y lo ejecutamos en un microprocesador con 4 núcleos de ejecución (cuádruple núcleo), como un Intel Core 2 Quad Q6700 o un AMD Phenom 9600, veremos que el uso neto del microprocesador no supera el 25% o el 27% si el sistema operativo está haciendo alguna tarea de fondo. Como sucedía en el caso anterior, si en la herramienta de monitoreo del microprocesador visualizamos un gráfico por cada núcleo o CPU (UCP), el planificador de procesos puede distorsionar los datos y confundirnos. Pues, al visualizar un gráfico por cada núcleo, es posible que veamos en el monitoreo que cada uno de los cuatro núcleos van teniendo una carga equivalente, pero ninguno de los cuatro llega casi al 100% en forma simultánea, situación que sería la ideal que buscaríamos para este bucle repetitivo. Es decir, otra vez parece que se fuera turnando el proceso. Pero, visualizando el uso neto del microprocesador, tenemos uno sostenido de entre el 25 y el 27% (ver la Figura 3) y tenemos cuatro núcleos, sin lugar a dudas, hay tres de ellos que están prácticamente con la totalidad de su potencia de procesamiento disponible, sin utilizar, pues la aplicación no los aprovecha para nada.

Como primera conclusión, podemos decir que a mayor cantidad de núcleos de procesamiento, más desperdicio de potencia sin utilizar. Mayor consumo de potencia, sin sentido, más generación de calor y un esfuerzo importante para disiparlo sin que éste se aproveche en mejoras para el rendimiento.

Tomando en cuenta que los fabricantes de microprocesadores planean seguir multiplicando la cantidad de núcleos de ejecución en los microprocesadores por venir, en la Tabla 1 sintetizamos el uso neto de la potencia de nuestro código de ejemplo ante diferentes cantidad de núcleos de procesamiento disponibles en un equipo teórico. Los números asustan y hacen ver que los desarrolladores debemos empezar a domar a este nuevo hardware en el corto plazo. Supongamos que hacemos la inversión para tener una estación de trabajo con dos microprocesador con cuatro núcleos cada uno de ellos, tendríamos un sistema de multiprocesamiento con ocho núcleos, un verdadero monstruo en términos estrictamente teóricos. Pero, semejante hardware sometido a la abundancia de código pensado para el monoprocesamiento, sería totalmente desperdiciado si las aplicaciones no están optimizadas para sacarle provecho. Pues, solamente se estaría utilizando el 12,50% de la capacidad instalada, por supuesto siempre si utilizamos únicamente esa aplicación.

En la semana en la cual comencé a escribir estas entregas, me tomé el trabajo de analizar una por una las aplicaciones que utilizo diariamente en las diferentes actividades que desarrollo con mi ordenador, equipado con un microprocesador





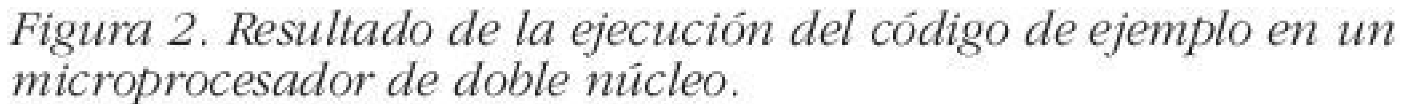

Figura 2. Resultado de la ejecución del código de ejemplo en un microprocesador de doble núcleo.

Figura 3. Resultado de la ejecución del código de ejemplo en un microprocesador de cuádruple núcleo (viéndolo como un solo gráfico para todos los núcleos).

Intel Core 2 Quad Q6700, con 4 núcleos. A pesar que trabajo en diferentes plataformas, todas ellas actualizadas con prácticamente las últimas versiones de los productos de software que se encuentran disponibles, el 80% de las aplicaciones utilizan únicamente 1 de los 4 núcleos de procesamiento disponibles, incluyendo las herramientas de desarrollo de software de última generación, a las cuales, muchas veces debo esperar un tiempo razonable para que terminen de compilar un desarrollo.

Éste es un buen ejercicio para llevar a cabo. Pues, allí nos damos cuenta de la inmensa necesidad que tiene en estos momentos la industria del software y de las soluciones informáticas de cambiar de paradigma de cara al futuro. Entonces, manos a la obra, vamos a sacar provecho de cuanto núcleo de procesamiento se encuentre disponible en el hardware en el cual se corran nuestras aplicaciones, al menos, en los procesos más críticos. Hay que tener sentido común, el esfuerzo para aprovechar los múltiples núcleos es importante, por lo cual, aplicarlo a absolutamente todo el código sería desmedido, pero, podemos hacerlo con las partes más críticas para comenzar.

¿De cuántos núcleos estamos hablando?

Para sacar provecho de los núcleos de procesamiento disponibles, deberíamos generar tantos hilos de ejecución *(threads)* como núcleos existan. Otra alternativa es realizar una autodetección para obtener un máximo, pero también permitir que el usuario pueda escoger un valor menor. Por ejemplo, si detectamos que el ordenador tiene 4 núcleos, puede ser que el usuario desee que la aplicación solamente utilice como máximo 2 núcleos, porque va a estar realizando otras tareas mientras deja procesando a nuestra aplicación.

En el Listado 2 se presenta código de ejemplo en los tres lenguajes de programación para poder detectar la cantidad de núcleos de ejecución disponibles y, en base a ello, tener una línea de base para que un mismo código consiga la máxima eficiencia tanto en un microprocesador con 1; 2 y 4 núcleos, o bien, en equipos con 8 y 16 núcleos dispersos en varios microprocesadores físicos.

Si bien tanto en .Net (Visual Basic y C#) como en Java, las propiedades o métodos utilizados hacen referencia al nombre Processor y Processors (Procesador y procesadores), en realidad, lo que nos devuelven es la cantidad de núcleos de procesamiento y no la cantidad de microprocesadores físicos. Por lo cual, por ejemplo, si ejecutamos este código en un equipo con un microprocesador Intel Core 2 Duo E6750, de doble núcleo, la variable miCantidadDeProcesadores tendrá el número 2, a pesar que se trata de un único microprocesador físico. Si lo hacemos en un microprocesador Intel Core 2 Quad Q6700, de cuádruple núcleo, la variable tendrá el número 4.

Tabla 1. Aprovechamiento de la potencia total de procesamiento disponible del código de ejemplo del listado 1, según la cantidad de núcleos disponibles

Cantidad de núcleos	Aprovechamiento aproximado
1	100,00%
2	50,00%
3	33,33%
4	25,00%
6	16,33%
8	12,50%
12	8,33%
16	6,25%
24	4,16%
32	3,13%
64	1,56%

A la hora de optimizar una aplicación para el multiprocesamiento, no tendría ningún sentido hacer todo el esfuerzo necesario para ello tomando como base una cantidad fija de núcleos de procesamiento. Pues, justamente el principal atractivo del paralelismo es la gran capacidad de escalabilidad presente y futura que le estaríamos brindando a cada uno de los procesos optimizados.

Una aplicación programada con una arquitectura de monoprocesamiento solamente es escalable si se mejora el hardware subyacente, es decir, si se agrega capacidad de ejecutar mayor cantidad de instrucciones por ciclo de reloj reemplazando el microprocesador por uno con una micro-arquitectura más optimizada o bien si se lo hace por uno con mayor frecuencia de trabajo. Como vimos con el ejemplo anterior, por más que agreguemos núcleos de procesamiento, éstos no ayudarán en nada a mejorar el rendimiento. Por lo cual, dadas las condiciones del mercado de los microprocesadores de la actualidad, en la cual se apunta al paralelismo, la escalabilidad de las aplicaciones con arquitecturas de monoprocesamiento están extremadamente limitadas.

En cambio, al diseñar los procesos pensando en un multiprocesamiento variable, es decir, cuyas tareas se dividen en tantos hilos de ejecución *(threads)* como núcleos de procesamiento dis-



LISTADO 2 Códigos de ejemplo para detectar la cantidad de núcleos de ejecución disponibles en el equipo, en el momento de ejecución (run-time)

```
Lenguaje de Programación Visual Basic:
' Utilizar Imports System.Environment
Dim miCantidadDeProcesadores As Integer
miCantidadDeProcesadores = Environment.ProcessorCount
Lenguaje de Programación C#:
// Utilizar Using System;
int miCantidadDeProcesadores;
miCantidadDeProcesadores = Environment.ProcessorCount;
Lenguaje de Programación Java:
int miCantidadDeProcesadores;
miCantidadDeProcesadores = Runtime.getRuntime().availableProcessors();
```

ponibles tenga el equipo, podemos garantizar una escalabilidad limitada únicamente por la cantidad de núcleos de procesamiento que presente el equipo. Por lo cual, si el proceso no consigue un rendimiento aceptable en un microprocesador de doble núcleo, migrando a uno con cuádruple núcleo, obtendremos una ventaja significativa en el rendimiento que de otro modo sería imposible de escalar hasta que aparezcan microprocesadores con nuevas micro-arquitecturas que permitan una ejecución de una mayor cantidad de instrucciones por ciclo.

Es por ello que, en cuanto se hace el esfuerzo para llevar a cabo el cambio de paradigma para desarrollar pensando en el multiprocesamiento y en el paralelismo, lo más conveniente es que existan mecanismos dinámicos en las aplicaciones para poder adaptar el proceso a una cantidad de núcleos de procesamiento determinado. De este modo, esa misma base servirá para que ese proceso se pueda ejecutar tanto en un equipo monoprocesador como en uno con 32 o más núcleos de ejecución, desde el mismo código y sin tener que hacer modificaciones adicionales. Pues, todo indica que el paralelismo seguirá creciendo con fuerza en los próximos años.

El primer paso para poder hacer que la optimización de los procesos más críticos de una aplicación para el multiprocesamiento en forma automatizada se pueda llevar a cabo era saber la cantidad de núcleos de procesamiento disponibles en tiempo de ejecución, lo cual conseguimos con el código del Listado 2.

Así como las aplicaciones multiplataforma se pueden ejecutar sin modificaciones en el código en los diversos sistemas operativos y plataformas para los cuales están validadas, podemos hablar de una nueva generación de aplicaciones aptas para el multiprocesamiento escalable. Pues, están disponible para ejecutarse desde equipos con monoprocesamiento hasta monstruos del paralelismo que hoy tal vez sólo podemos imaginar y en pocos años serán una realidad en muchos escritorios (así ha pasado siempre en el cambiante e impredecible mercado tecnológico).

El problema del GC

El viejo y conocido refrán es muy claro "no todo lo que brilla es oro" y, sin lugar a dudas, lo podríamos aplicar perfectamente a la programación con múltiples hilos de ejecución. Los lenguajes de programación de alto nivel modernos, como los tres que estamos utilizando para los ejemplos (Visual Basic; C# y Java) utilizan mecanismos automáticos de liberación de memoria que impactan fuertemente en el rendimiento de los diferentes hilos.

Cuando declaramos una variable, ya ocupa un lugar en memoria. En algunos casos, cuando se le asigna un valor, requiere aún más espacio en memoria.

En algunos lenguajes de programación (como C y C++), es necesario liberar el espacio de memoria utilizado por las diferentes variables cuando no se usan más, pues sino quedarán asignados sin emplear y cada vez quedará menos cantidad de memoria disponible para las aplicaciones en ejecución.

Esto presenta bastantes inconvenientes, pues si nos equivocamos en liberar la memoria de una variable que pensábamos que no se utilizaría más y luego se sigue usando, se producirán errores durante la ejecución del programa.

Para evitar los grandes dolores de cabeza que trae a los programadores la liberación de la memoria utilizada por las variables, estos tres lenguajes analizados incorpora un mecanismo automático para liberar la memoria de las variables que ya no se emplean. Por lo tanto, no es necesario preocuparse por este tema, lo cual, sin lugar a dudas, es una gran ventaja. De esta manera podemos concentrarnos mucho más en lo que queremos hacer y no tener que andar pensando en cuándo debemos liberar la memoria de una variable determinada. Además, el código de los programas resulta mucho más fácil de leer y comprender, pues no queda sucio por las liberaciones de memoria.

Estos sistemas que incorporan se conocen con el nombre de recolección de basura *(garbage collection)* o bien por sus siglas GC. Pues, un proceso invisible para el desarrollador se encarga de juntar a todas las variables que ya no se están utilizando más y libera toda la memoria que consumían.

Sin embargo, a la hora de desarrollar utilizando múltiples hilos de ejecución nos encontramos con un gran inconveniente. La recolección de basura generalmente se lleva a cabo en un único hilo de ejecución que puede degradar el rendimiento de cada uno de los hilos independientes que con tanto esfuerzo se habían construido para conseguir el mayor aprovechamiento del hardware.

Claro, en C y C++ como el programador tiene el control exacto de las liberaciones de memoria y no hay un proceso de recolección de basura que puede ir cortando el rendimiento al frenar a los hilos de ejecución independientes hasta que este proceso culmine, cada cierta cantidad de tiempo, se puede conseguir una mayor independencia del comportamiento de ese proceso. Sin embargo, nos sirve a modo de comparación únicamente, pues nuestros objetivos se centran en los tres lenguajes de programación mencionados.

El comportamiento del recolector de basura dependerá exclusivamente del tipo de proceso que estamos preparando para el multiprocesamiento, por lo cual, las diferentes técnicas para evitar que su funcionamiento degrade el rendimiento del proceso dependerá exclusivamente de realizar pruebas y en base a ellas tomar determinaciones que hagan que su comportamiento sea más predecible.

Existen numerosas investigaciones presentadas por los mismos desarrolladores de los lenguajes de programación y de sus entornos de desarrollo que muestran cómo el afinamiento erróneo del comportamiento del recolector de basura incide negativamente en la concurrencia y en la escalabilidad, y por ende, en las aplicaciones que utilizan multiprocesamiento.

En la actualidad, se trata de uno de los primeros cuellos de botella *(bottlenecks)* con los cuales se enfrenta el desarrollador cuando comienza a separar sus procesos en múltiples hilos de ejecución.

En .Net (Visual Basic y C#), podemos utilizar la variable GC, la cual contiene una instancia de la clase GC (*Garbage Collector* – Recolector de basura). Utilizando diferentes métodos, podemos optimizar el rendimiento de la aplicación cuando conocemos cuál será el comportamiento del recolector de basura que más beneficie a un proceso determinado. Pues, en tiempo de ejecución, el entorno no sabe exactamente qué vamos a hacer con una variable determinada más adelante, por lo cual, por más optimizacio-



nes que lleve a cabo el compilador, es el desarrollador el que mejor conocimiento tiene para mejorar el control del recolector.

Llamando al método GC.Collect(), sin ningún parámetro, se forzará la ejecución del recolector de basura para todas las generaciones de variables, produciendo la barrida más completa. Hacer un llamado manual a este método en medio de un lazo o bucle dentro de un hilo de ejecución (thread) sin lugar a dudas afectará al buen rendimiento de este hilo y puede hacerlo a nivel global en los otros que se están ejecutando.

Ahora bien, en los ejemplos que vamos a ir desarrollando, no va a haber un historial previo del recolector de basura, pues arrancarán directamente lanzando los múltiples hilos de ejecución. Sin embargo, en una aplicación real, esos procesos especialmente optimizados para el multiprocesamiento pueden arrancar luego de que hayan corrido otros procesos. En tal caso, una buena medida que puede ayudar a disminuir los efectos negativos de una recolección lanzada automáticamente por el entorno de ejecución en medio de la ejecución de los hilos, es conveniente, antes de arrancar la creación de los múltiples hilos y de lanzar los lazos en cada uno de ellos, es llamar a una recolección de toda la basura que se generó hasta ese momento como primera instrucción, a través del método GC.Collect(), sin parámetros.

Otro método muy interesante es GC.KeepAlive (Mantener vivo), el cual recibe como parámetro un tipo Object, es decir, la referencia a un objeto que queremos que no sea tenido en cuenta por el analizador previo al llamado de recolección de basura y, de esta manera, no insuma recursos. Esto no significa que no se vaya a recolectar cuando se deje de utilizar, sino que por el momento no se lo tenga en cuenta.

También, puede resultar interesante llamar a la recolección de basura para la generación a la cual pertenece un objeto en particular, pero, solamente para ésta. Pues, en algunas ocasiones determinados procesos pueden consumir muchos recursos durante su ejecución, los cuales pueden generar luego una recolección demasiado pesada que se note en otro momento. Tener el control de los momentos puede resultar muy interesante en .Net cuando trabajamos con concurrencia (múltiples hilos).

Para llevar a cabo una recolección de basura de una generación en particular, podemos utilizar el método GC.Collect, pasándole como parámetro el número de generación que debe recolectar y, en forma opcional, un segundo parámetro con el modo de recolección. Para saber la generación a la cual pertenece un objeto determinado, podemos llamar al método GC.GetGeneration y pasarle como parámetro la variable que hace referencia a cualquier instancia de una clase derivada de Object.

El modo de recolección puede ser alguno de los siguientes tres:

- GCCollectionMode.Default. El predeterminado.
- GCCollectionMode.Forced. Forzar al máximo la recolección.
- GCCollectionMode.Optimized. Optimizar para obtener la mejor relación entre el rendimiento y el peso de la recolección.

Por ejemplo, para llamar a la recolección forzosa de la generación correspondiente a la variable loCirculo, se puede hacer con la siguiente línea de código:

Lenguaje de Programación Visual Basic:

```
GC.Collect(GC.GetGeneration(loCirculo),
GCCollectionMode.Optimized)
```

Lenguaje de Programación C#:

```
GC.Collect(GC.GetGeneration(loCirculo),
GCCollectionMode.Optimized);
```

En Java, la situación es un tanto diferente, pues no tenemos un control tan completo en tiempo de ejecución del comportamiento del recolector de basura a través de código. En cambio, sus parámetros de funcionamiento se establecen antes de la ejecución de la aplicación, indicándole el comportamiento a la JVM (Java Virtual Machine – Máquina virtual de Java).

Llamando al método System.gc() o bien a su equivalente Runtime.getRuntime().gc(), sin ningún parámetro, se forzará la ejecución del recolector de basura para todas las variables, produciendo la barrida más completa. Al igual que explicamos para los dos lenguajes .Net, hacer un llamado manual a este método en medio de un lazo o bucle dentro de un hilo de ejecución (thread) en Java afectará al buen rendimiento de este hilo y puede hacerlo a nivel global en los otros que se están ejecutando.

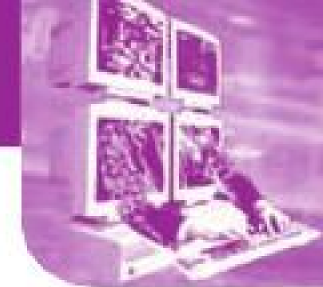
Pero, también resulta una buena práctica que puede ayudar a disminuir los efectos negativos de una recolección lanzada automáticamente por la JVM en medio de la ejecución de los hilos, hacer una llamada a una recolección de toda la basura que se generó hasta ese momento como primera instrucción, antes de arrancar la creación de los múltiples hilos y de lanzar los lazos en cada uno de ellos.

Ejecutando una aplicación java desde la línea de comandos con la opción -verbose:gc, obtendremos estadísticas de las veces que se llevó a cabo la recolección de basura de las diferentes colecciones (el equivalente a las generaciones en el mundo de .Net). Estas recolecciones generan una pausa en la ejecución de los hilos que generalmente se siente en procesos con muchos hilos corriendo simultáneamente y que hacen un procesamiento con grandes volúmenes de datos, pues el tiempo puede ser superior a 1 segundo, lo cual afecta al rendimiento global de la aplicación. Sin embargo, generalmente la interpretación de estos datos puede llevar a confusiones si no estamos acostumbrados a trabajar con estas bitácoras, por lo cual, es más conveniente utilizar un *Profiler*, como por ejemplo, el que se incluye en NetBeans IDE 6.0. Éste nos permite monitorear los diferentes hilos de ejecución y el tiempo total de ejecución que se está utilizando en el recolector de basura (GC), por ejemplo en la Figura 4 podemos ver cómo nos está indicando que en esa aplicación, con varios hilos de ejecución con fuerte demanda de procesamiento, está consumiendo un 5,4% del tiempo total de procesamiento.

En este punto es muy importante hacer una aclaración. Como la JVM es multiplataforma, muchas veces los resultados del recolector de basura ante el multiprocesamiento serán diferentes en las diversas plataformas en las cuales está disponible y en las diferentes versiones de Java se ha trabajado mucho en mejorar los algoritmos para adaptar el recolector de basura dinámicamente en tiempo de ejecución al tipo de aplicación que se está ejecutando, a la carga que conlleva y a la plataforma de hardware sobre la cual está montada la JVM. Por lo cual, no hay que sorprendernos si los resultados son muy diferentes de una plataforma a otra, inclusive, ante cambios en el hardware subyacente.

También debemos tener en cuenta que el Profiler por supuesto agrega una sobrecarga de procesamiento (overhead) a la aplicación. Sin embargo, es la mejor forma que tendremos de visualizar el comportamiento de la JVM y realizar ajustes en el heap size (tamaño del heap) acorde a los resultados de la aplicación y agregar las llamadas a recolecciones de basura por código para generar el mejor entorno posible para que los hilos de ejecución consigan el mejor rendimiento posible del hardware instalado.

En la Figura 5 podemos ver otro ejemplo de varios hilos de ejecución que trabajan fuertemente con concatenación de cadenas de caracteres (String), lo cual hace que el recolector de basura tenga demasiado trabajo y esté consumiendo un 89,2% del tiempo total de procesamiento. Inaceptable, sin lugar a dudas, pues



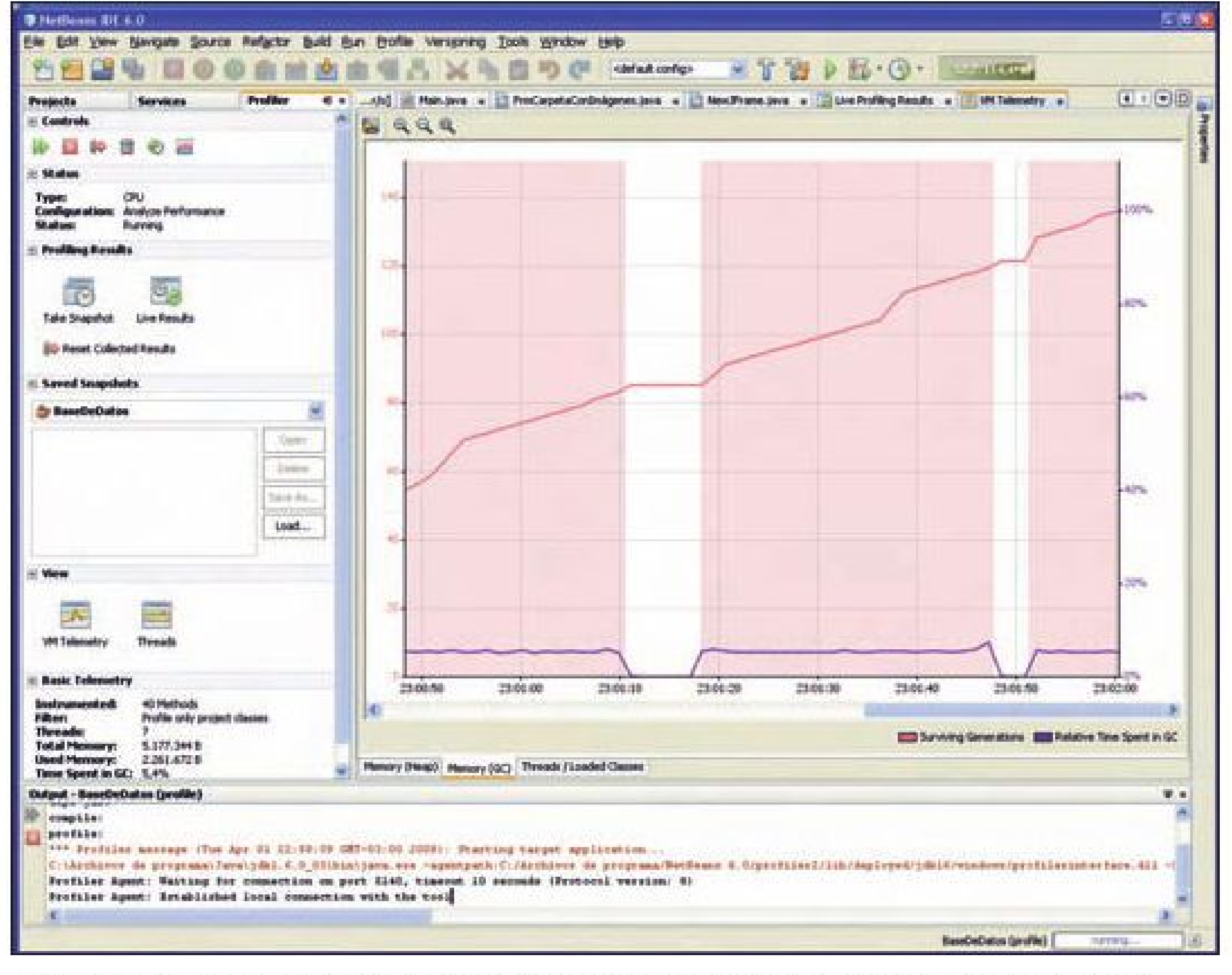


Figura 4. El Profiler de NetBeans IDE 6.0 mostrando el tiempo total de procesamiento consumido por el recolector de basura (Time spent in GC) y el gráfico de la relación de memoria.

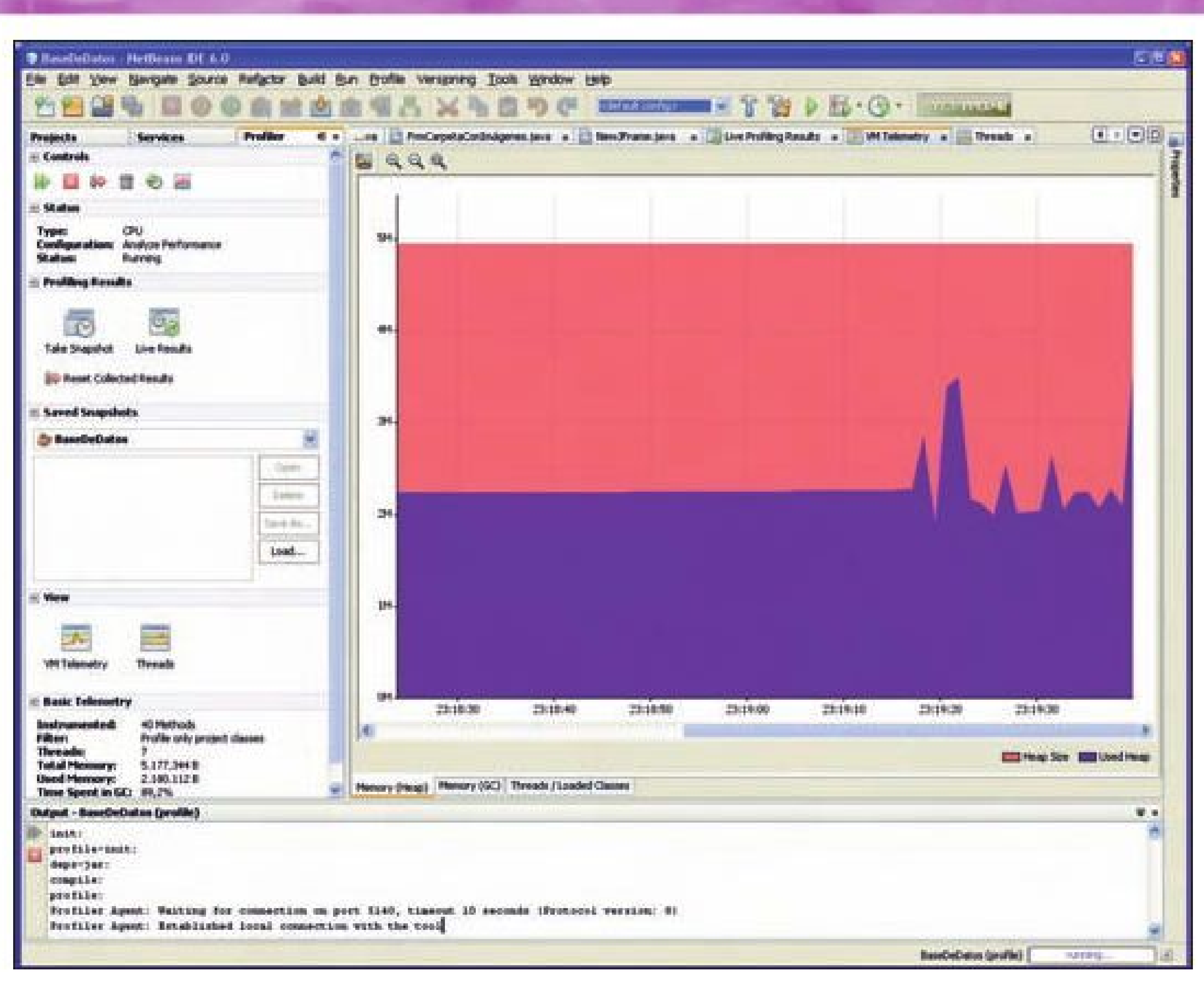


Figura 5. El Profiler de NetBeans IDE 6.0 mostrando un caso de tiempo total de procesamiento consumido por el recolector de basura (Time spent in GC) de 89,2% y el gráfico de la relación de memoria.

solamente se esta aprovechando un 10% de la capacidad total de procesamiento. Como podemos ver, el trabajo con cadenas de caracteres en los lenguajes de programación modernos, en los cuales, como en Java, son objetos en vez de un simple tipo de datos, hacen que el recolector de basura tenga mucho trabajo y consuma un tiempo de procesamiento excesivo.

Ante esa situación, el aumento del rendimiento por la separación en hilos de ejecución se ve totalmente opacado por un excesivo trabajo del recolector de basura. Es por ello que hay que prestar especial atención al tiempo de procesamiento que consume y optimizar los hilos de ejecución y el tamaño del heap para armonizar el rendimiento global.

Por ello, decíamos que no todo lo que brilla es oro y tenemos que trabajar con especial cuidado para no equivocarnos a la hora de realizar la separación de los procesos. Todo esto se debe tener en cuenta.

Cuando los hilos no alcanzan, los procesos no fallan

Siempre hay que tener un as bajo la manga. Los desarrolladores sabemos que las mismas soluciones que funcionan para un caso pueden ser muy poco eficientes en otros. Por lo cual, es bueno saber que si no encontramos la vuelta para poder conseguir un mayor rendimiento a través de los hilos de ejecución, los cuales, sin lugar a dudas deben ser la primera opción que debemos intentar, podemos recurrir a los procesos independientes.

Si vamos a ver en detalle todo lo necesario para aplicar hilos de ejecución exitosamente en Visual Basic, C# y Java, ¿por qué presentamos otra alternativa? La respuesta es sencilla, debido a que la implementación de la concurrencia y de los múltiples hilos de ejecución en estos tres lenguajes de programación, aún en sus versiones más modernas y recientes, no es perfecta y todavía existen muchos temas para resolver para conseguir el paralelismo más eficiente posible acorde a las arquitecturas de hardware modernas. Esto hace que, ante determinadas ocasiones, debemos tener otra alternativa.

Si tomamos el ejemplo del Listado 1, el cual estaba diseñado sin múltiples hilos de ejecución, lo compilamos y lanzamos 4 instancias de esa aplicación casi en simultáneo, en un equipo con un microprocesador con 4 núcleos de ejecución, veremos que efectivamente lleva al 100% al consumo del procesador, sin ninguna modificación al código. Claro, en ese caso estamos creando 4 procesos totalmente independientes, a los cuales el planificador del sistema operativo asignará uno a cada núcleo de procesamiento disponible.

Si lográramos un mecanismo de coordinación que permitiera que cada uno de estos procesos tuviera la información para comenzar y finalizar una parte determinada de un proceso global, conseguiríamos fácilmente y sin involucrarnos en toda la problemática de los múltiples hilos de ejecución, mejorar un proceso para que aproveche las capacidades de paralelismo del hardware. Nos olvidaríamos de las imperfecciones de la concurrencia del lenguaje de programación, de los inconvenientes del recolector de basura y de muchos otros temas que todavía no hemos analizado en detalle.

Es muy sencillo lanzar aplicaciones y pasarles parámetros en cualquiera de los tres lenguajes de programación. Una aplicación puede lanzar tantas como núcleos de procesamiento haya disponibles e indicarle en dónde debe empezar y hasta dónde debe procesar. Así tenemos un esquema de un coordinador que monitorea a los n procesos independientes que se generan. Puede ser una solución ante determinadas tareas pesadas que tienen una complejidad tan grande que hace que su división en hilos de ejecución no nos lleve a resultados óptimos o que tome demasiado tiempo esa optimización para el multiprocesamiento.

Muchos procesos por lotes *(batch)* extremadamente complejos, con requerimientos de muchos accesos a disco y a bases de datos en cada iteración pueden hacer fracasar cualquier intento de paralelización a través de hilos de ejecución si no tenemos cierta experiencia y práctica en el tema. Pero, la separación en procesos puede resultar una variante mucho más sencilla de implementar que hará que tengamos un rápido éxito en la optimización de ciertas tareas para el multiprocesamiento, hasta que tengamos el conocimiento necesario para manejar con cuidado los múltiples hilos y la concurrencia.

Por lo tanto, no hay que descartar este esquema, en el cual en vez de tener una coordinación en mismo código en el cual se lanzan los múltiples hilos, tenemos una aplicación que lanza a otras con parámetros y las monitorea, por ejemplo, a través de reportes de progresos mediante sistemas de comunicaciones complejos o bien simplemente a través de la grabación de información de seguimiento en tablas de una base de datos relacional.



Conclusión

El uso de múltiples hilos *(threads)* en Visual Basic; C# y Java es complejo y trae aparejadas nuevas problemáticas y conceptos que antes no debíamos tener en cuenta. Sin embargo, eleva la potencia de procesamiento que podemos brindar a las aplicaciones.
Lo principal es comprender los nuevos paradigmas de desarrollo y saber cómo aprovecharlos. Hace unos once años, en 1997, tuve la oportunidad de presenciar la presentación de la primera versión de JBuilder, de la empresa que hace poco volvió a ser Borland, en un evento realizado en EE.UU. En uno de los tantos cafés pude dialogar un buen rato con uno de los principales ingenieros del grupo de trabajo sobre ciertos conceptos de mi interés en aquel entonces de la persistencia de los objetos en las bases de datos y el enfoque que le estaban dando y al que apuntarían en futuras versiones. La respuesta me quedó grabada ante una de las preguntas que le había hecho: "soy un ingeniero, pero tengo que ser pragmático porque sino no duermo nunca". El pragmatismo también se debe aplicar al desarrollo con múltiples hilos de ejecución, pues sino uno nunca terminaría de optimizar una aplicación para el multiprocesamiento.
En algunos casos, serán múltiples hilos con un manejo inteligente del recolector de basura, en otros, se lanzarán múltiples procesos y se coordinarán sus ejecuciones y progresos. Todo es válido mientras se respeten ciertas reglas y se consigan los objetivos consiguiendo un costo de mantenimiento razonable. Dependerá de cada caso.
En esta entrega hemos ofrecido una detallada introducción a las necesidades del uso de múltiples hilos *(threads)* en Visual Basic; C# y Java; así como a los conceptos fundamentales y los principales enemigos y problemas con los cuales nos encontraremos en este largo pero apasionante camino. En las siguientes entregas vamos a desarrollar ejemplos completos con las técnicas para desmenuzar un proceso en múltiples hilos, dinámicos según el hardware subyacente, a coordinarlos y a resolver los problemas más importantes con los cuales nos encontraremos en el 95% de los casos. Veremos cómo ayudar al planificador del sistema operativo a no equivocarse y a administrar las prioridades entre los diferentes hilos de ejecución. Todos los casos son tomados en base a experiencias de la vida real de cualquier desarrollador moderno con el desafío de explotar al máximo el paralelismo y conseguir que una aplicación por sí misma muestre un gráfico como el de la Figura 6. SP

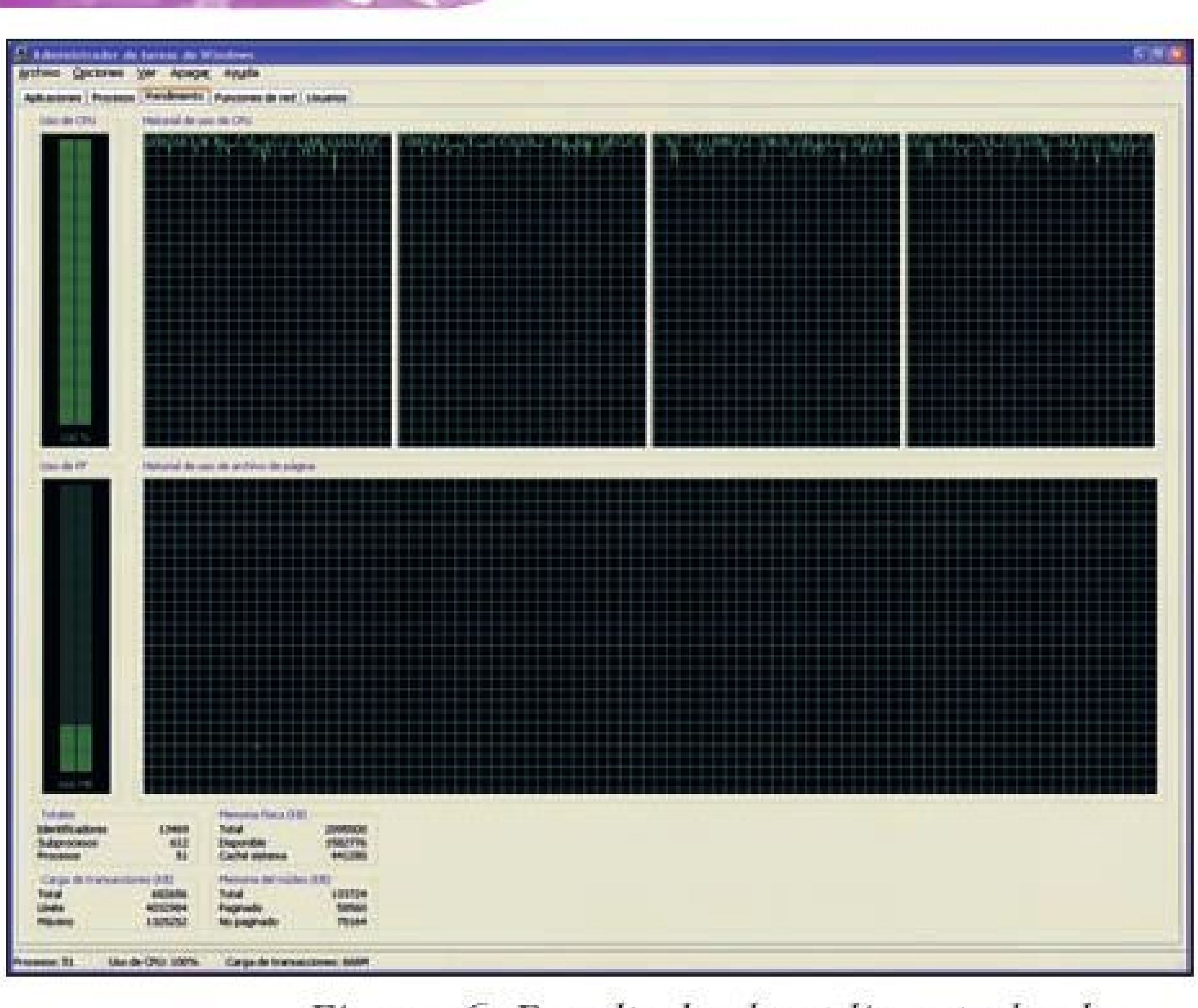
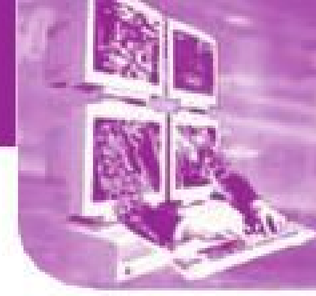


Figura 6. Resultado de aplicar todos los conocimientos de estas entregas en un proceso crítico de una aplicación en un microprocesador de cuádruple núcleo.



Microsoft ASP.NET MVC (paradigmna de modelo, controlador y vista)

ERICH BUHLER

El paradigma de modelo, controlador y vista de Microsoft ASP.NET (MVC) es una nueva implementación compatible con Visual Studio, la que ofrece una solución diferente para la construcción de aplicaciones Web. En este artículo analizaré sus ventajas, así como sus parecidos con ASP.NET.

Introducción

Hoy me encargaré de explicarte todo sobre el paradigma de modelo, controlador y vista (MVC en inglés), el que en sí ofrece una aproximación diferente para el desarrollo de aplicaciones Web. Éste viene siendo usado por empresas de código abierto desde hace ya bastante tiempo (ej. Monorail), pero sin embargo, es la primera vez que Microsoft incursiona en el terreno. Como verás más adelante, el artículo no solamente te servirá para aprender esta tecnología, sino que también te mostraré otras que son ampliamente usadas, lo que finalmente te facilitará el camino para comprender lo que se viene.

Figura 1

Material adicional

El material complementario puede ser descargado desde nuestra web www.revistasprofesionales.com

En realidad, todo comenzó el año pasado, cuando Scott Guthrie (encargado general de la infraestructura .NET) se encontraba en un viaje de negocios, y particularmente dentro del avión de una importante aerolíneas. Es así que vaya a saber porqué (quizá debido a que la revista de abordo no ofrecía entretenimiento adecuado o ya había sido leída en reiteradas ocasiones) que tomó la decisión de implementar un pequeño ejemplo, para poder así ver si el modelo de vistas y controladores era algo viable en ASP.NET. Debido al contexto donde todo ello aconteció, las primeras versiones fueron bastante básicas, aunque no por ello demostraban la factibilidad de éste en el entorno de Microsoft. Desde allí hasta esta parte, muchas cosas han cambiado, pero la esencia sigue siendo la misma... contar con una infraestructura que permita resolver las aplicaciones Web de una forma diferente.

"Las primeras versiones fueron básicas, pero ahora ya se cuenta con una infraestructura bastante completa de modelo y controlador y vista (MVC)"

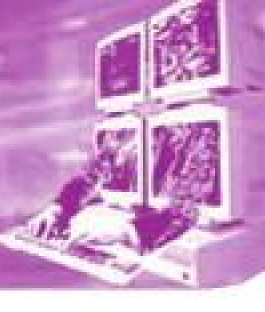
Cuando te cuente más adelante de que se trata el modelo y los controlador y vista, verás que la aproximación es de extrema utilidad para un conjunto particular de aplicaciones, pero no se debe tomar como un reemplazo total de ASP.NET. Este último seguirá existiendo, pero ahora, se tendrás mas opciones a la hora de crear una solución Web.

Por donde comenzar...

Antes de zambullirte de lleno en la nueva arquitectura de Microsoft, necesitarás bajar e instalar la misma, así como también contar con Visual Studio 2008; lo que implica implícitamente la existencia de .NET 3.5 en tu sistema. Ve entonces al siguiente hipervínculo para obtener el paquete:

```
http://www.asp.net/downloads/
3.5-extensions/
```

(ver Figura 1)



Es importante que sepas que ASP.NET MVC no puede ser instalado en la edición "express" de Visual Web Developer 2008. Si todo sale bien, deberías encontrar el siguiente nuevo proyecto al abrir Visual studio.



Figura 2.

"ASP.NET MVC por ahora funciona en las versiones no 'express' de Visual Studio"

La instalación agrega algunos nuevos tipos de archivos, los que te mostraré más adelante. Es tiempo entonces de que veamos la razón de ASP.NET MVC, antes de conocer concretamente de que se trata.

Ser o no ser, esa es la pregunta

Cuando se construye una aplicación Web en ASP.NET, el paradigma que se emplea se basa principalmente en formularios, controles y eventos, los que son iniciados mediante una técnica en HTTP llamada Post. Esto es en principio la aproximación adecuada, y de hecho, hace que sea sencillo implementar la interfaz gráfica y su código asociado, debido a la forma en la que toda la parafernalia funciona. En definitiva, la aproximación es parecida a la que encuentras en el mundo de ventanas de Windows. Cuando digo esto, lo hago desde el punto de vista funcional, ya que como sabrás, hay muchas diferencias entre un mundo y el otro. No obstante, dejaré de lado esto para centrarme en la razón por la cual existe ASP.NET y posteriormente en el porqué de ASP.NET MVC (modelo, controlador y vistas). Es entonces que un formulario Web brinda las siguientes 3 características:

- Es relativamente fácil de crear aplicaciones Web mediante formularios
- Es posible usar directamente propiedades y métodos de controles desde los procedimientos de eventos, de la misma forma que lo hacías en los proyectos de formularios Windows
- Es relativamente fácil el rehusar un formulario Web

Ahora bien, con algunos tipos de aplicación el resultado puede ser un tanto engañoso sino nefasto. Cuando hago esta aseveración me refiero exclusivamente a aquellas soluciones que están fuertemente atadas a una base de datos y que quizá emplean alguna lógica de negocio (pero no al revés). Para ver el problema, presta atención a la figura 3, la que muestra la información de la tabla de socios de un gimnasio.



Figura 3.

Hasta aquí todo va bien, y de hecho el usar un control de tabla se presenta como el candidato ideal. No obstante, días más tarde te encuentras trabajando como de costumbre, cuando Adrian, que es uno de los desarrolladores de tu equipo, te pide permiso para invocar la pantalla que lista socios desde otra parte de la solución. La idea es simplemente ofrecer acceso al formulario desde otro lugar. Tú inmediatamente aceptas, ya que en definitiva, te sientes orgulloso de tu implementación. Una semana más tarde, Adrian te pregunta si es posible que la información de la tabla pueda ser filtrada por nombre de socio, a lo cual accedes sin problema. Tu idea para solucionar esto es la de pasar un parámetro en la línea de dirección URL (URL extendido), como se muestra a continuación:

```
Lista.aspx?FiltroNombre=Carlos
```

Para el caso de que nada se especifique, entonces se mostrarán todas las filas. Un mes después, otro desarrollador de tu equipo requiere el adicionar una característica que haga posible mostrar el nombre como hipervínculo, con el fin de enlazar éste con una pantalla externa que hace posible modificar la información del socio. Como tienes muy poco que hacer, simplemente adicionas un nuevo parámetro, el que indica si la funcionalidad estará o no disponible.

```
Lista.aspx?`FiltroNombre=
Carlos&MostrarHipervínculo=si
```

Por último, Andrea (que es la encargada del diseño gráfico) te pregunta si podrías tener 2 versiones de la misma pantalla empleando diferentes estilos y tipos de letra. Es así que aquí aplicas una nueva estrategia, la que consiste en pasar el nombre de una hoja de estilos, la que deberá ser escrita y guardada en la misma carpeta del formulario Web.

```
Lista.aspx?FiltroNombre=
Carlos&MostrarHipervínculo=
si&NombreHojadeEstilo=versiónenVerde
```

Pese a que todo funciona correctamente, al cabo de unos meses ves como más y más parámetros son adicionadas a tu formulario Web, el que originalmente estaba simplemente pensado para exhibir la lista de socios. El resultado final comienza a verse minado de estructuras "if" que efectúan tareas condicionales (listado 1). Y como normalmente las aplicaciones siempre crecen a nivel en funcionalidad, es de esperar que la lista se incremente aún más a lo largo de los meses.

Pues entonces... ¿Puedes ver en dónde se encuentra el error de diseño?

El mismo radica en que estás utilizando una sola pantalla para representar comportamientos variados, sin importar finalmente si ellos se relacionan entre sí. Es decir, todas las instancias hacen referencia o comparten el mismo conjunto de datos (le llamaremos "Modelo" más adelante), pero esto no implica que la funcionalidad sea la misma. Por otra parte, no existe una opción clara cuando se quieren ofrecer distintas representaciones visuales para un mismo formulario Web.

"ASP.NET funciona correctamente, pero falla cuando se quiere comienzan a adicionar diferentes comportamientos para un mismo formulario Web"

Podrías pensar en construir 3 o 4 formularios Web, cada uno con su propia versión de la tabla de socios, pero ello implicaría replicar parte del código. Otra opción sería implementar un control Web, y posteriormente reusar el mismo en varios formularios. De vuelta has caído en la trampa ya que tu componente debería ser "super" inteligente". Es así que ASP.NET brinda una buena opción, pero no cuando se quiere que un formulario Web que está estrictamente asociado a datos brinde distintos comportamientos, así como tampoco tenga varias presentaciones gráficas. La siguiente lista exhibe un resumen de los inconvenientes que te encontrarás cuando una aplicación que está fuertemente enlazada a una base de datos se expone a través de formularios Web ASP.NET:

- Se hace muy fácil la reutilizar un formulario Web, lo que trae como consecuencia que se terminen adicionando múltiples comportamientos (muchos de ellos no relacionados).
- No es fácil brindar distintas presentaciones gráficas para un mismo formulario.



LISTADO 1

```
if (Request.QueryString["FiltroNombre"] != null) {
    //Filtra por nombre
}

if (Request.QueryString["MostrarHipervínculo"] != null) {
    //Exhibe u oculta el hipervículo para editar
}
if (Request.QueryString["NombreHojadeEstilo"] != null) {
    //Cambia la hoja de estilo
}
```

- Si se desea poner a prueba los componentes de la aplicación (testing), se deben emplear herramientas que puedan lidiar con la interfaz gráfica Web, lo que hace que sea difícil de aislar la lógica de negocio del formulario.

Conociendo un poco más sobre MVC

Es sano que sepas que existen dos paradigmas del modelo de controladores y vistas (pasivo y activo), aunque me centraré solamente en el primero ya que es el empleado por Microsoft. Éste se basa en la separación física de la aplicación en 3 partes. Cada una de ellas tiene un role específico, el que básicamente ayuda a organizar sus funcionalidades y hace que los elementos sean reusables y fáciles de mantener.

- Modelo
  Este es el sitio donde se almacena el código que obtiene la información de la base de datos. La clase "Socios" con su método "obtener" debería ir allí, así como también los componentes que llevan adelante la lógica de negocio (por ejemplo, "calcular DeudaSocio").
- Vistas
  Este es el sitio donde las interfaces gráficas deberán ser depositadas. Las mismas no tienen contacto directo con las clases del modelo (base de datos o lógica de negocios). En sí, alguien (controlador) ejecutará la lógica de negocio y obtendrá la información, pasándola posteriormente a la vista (página).
- Controlador
  El controlador recibe una acción del usuario y decide que página (vista) deberá ser mostrada. El proceso implica invocar la lógica de negocios adecuada, obtener los datos, e inyectar la información en la vista, la que será finalmente será retornada al usuario. (ver Figura 4)

Como característica adicional, este paradigma emplea una arquitectura llamada REST (Representational State Transfer o estado figurativo de transferencia), la que básicamente indica como se organizan los recursos en una solución Web y como ellos son accedidos. Pese a la formalidad del nombre, verás en breve que esto no es más que sentido común (¡el menos común de todos los sentidos hoy en día!). He hecho una brevísima reseña de la forma en la que el paradigma funciona, ahora me centraré en como todo se relaciona con ASP.NET MVC.

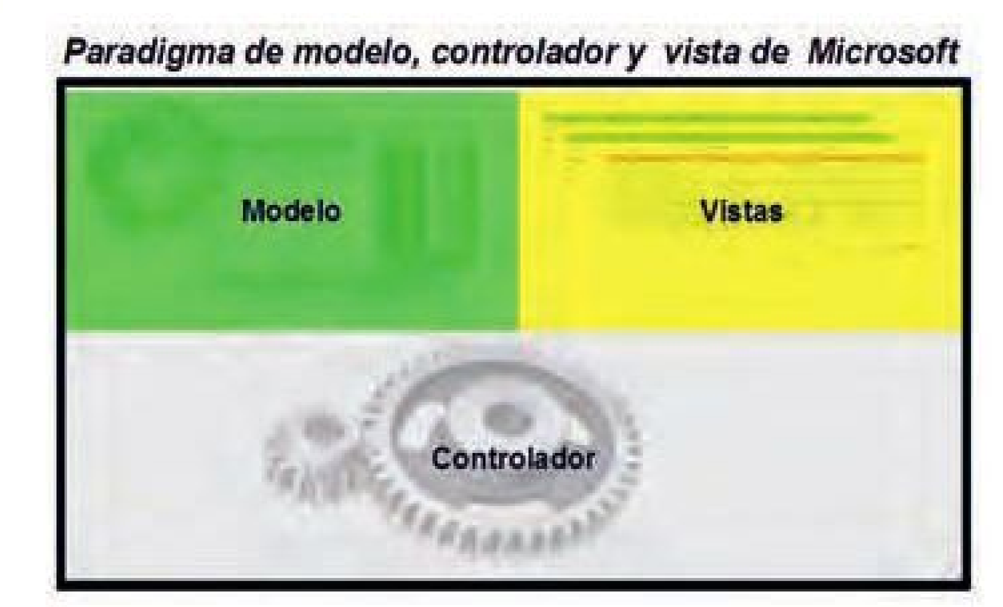


Figura 4.

La primera aplicación ASP.NET MVC

Para ver la tecnología en su conjunto, no hay mejor ejemplo que el que viene acompañado de una buena práctica, por lo que comenzaremos creando una nueva aplicación Web de tipo ASP.NET MVC (figura 5). Cada vez que se selecciona un proyecto de este tipo, Visual Studio adiciona un par de páginas que ofrecen un buen ejemplo. Pero antes de que lo analicemos, quiero que prestes atención a la estructura; fíjate las nuevas referencias, las que constituyen la primera diferencia (figura 6).

Estos nuevos ensamblados son el núcleo del modelo MVC en Microsoft ASP.NET; sin ellos nada de este mundo sería posible. A su vez, se crean 3 carpetas, las que almacenarán de forma separada los elementos pertenecientes al modelo, vista y controlador; observa la figura 7 (veremos en breve como que se relaciona cada uno, así como su utilidad y funcionamiento).

Todo lo que necesitas saber sobre REST

Hace unos meses atrás tuve una situación laboral que me puso entre la espada y la pared... me consultaron si podía definir que era REST y su funcionamiento. Tengo claro que hoy en día es una pregunta clave en las entrevistas de trabajo y sé que finalmente puede decidir entre el éxito o fracaso. Es entonces que quiero extenderme un poquito en este tema, para que no te queden dudas, y de paso ayudarte en tu vida laboral, así como también a comprender Microsoft ASP.NET MVC.

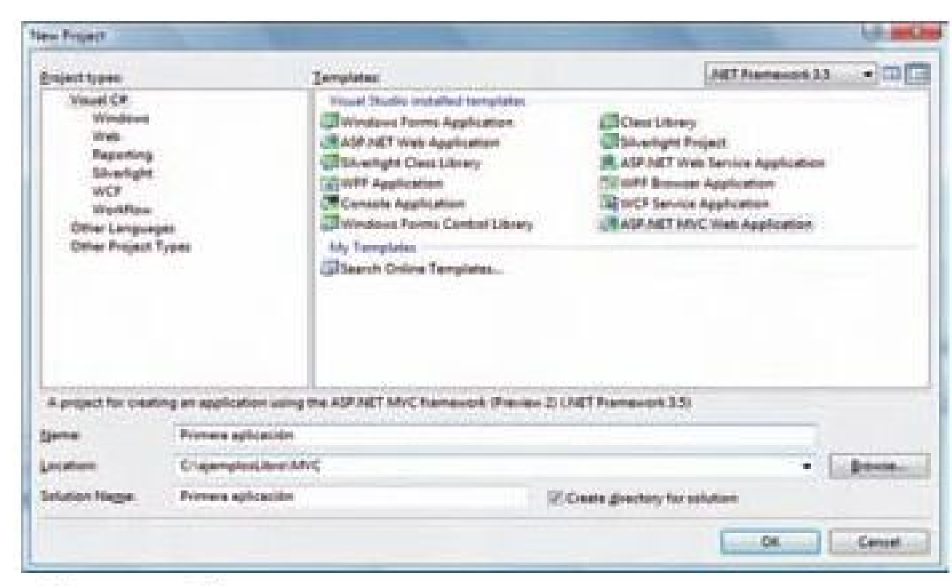


Figura 5.

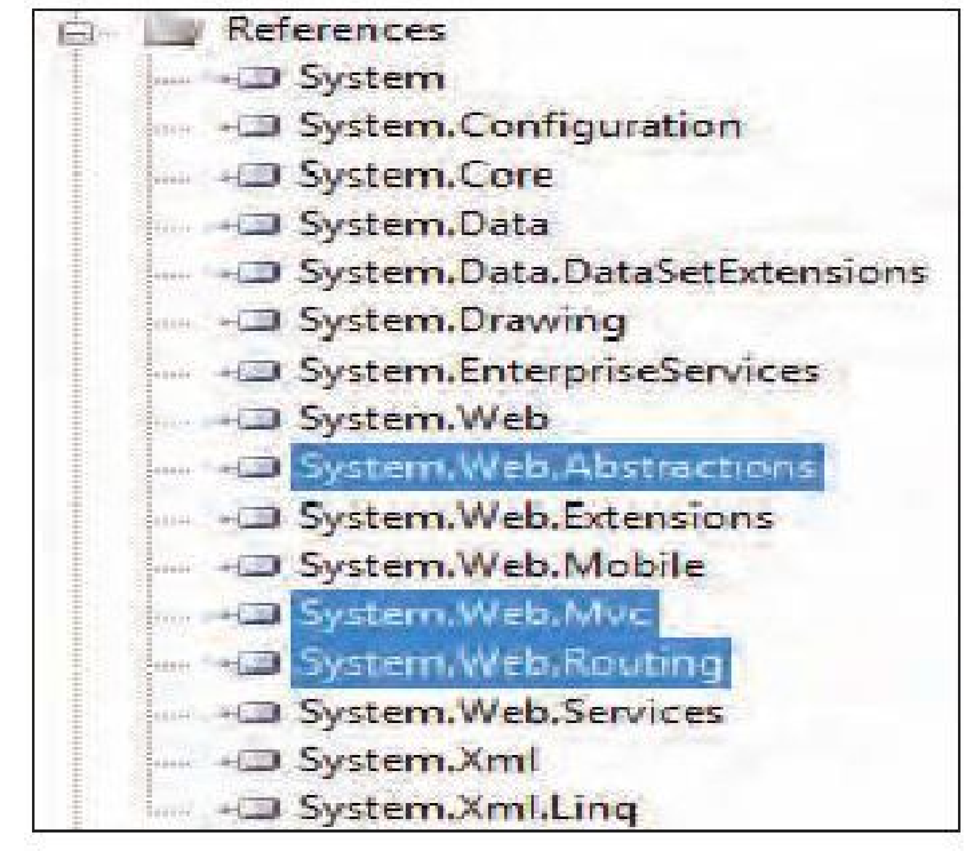


Figura 6.

Si ejecutas la aplicación creada previamente, verás que se abre una página predeterminada, la que ofrece dos fichas, una llamada "Home" y otra "About". Si haces clic en ellas se traerán las pantallas respectivas, en forma similar a lo que se haría en ASP.NET. (ver Figura 8)

No obstante, si eres un buen observador, habrás notado que en la barra de direcciones no se menciona el nombre de la página o recurso, simplemente la carpeta.

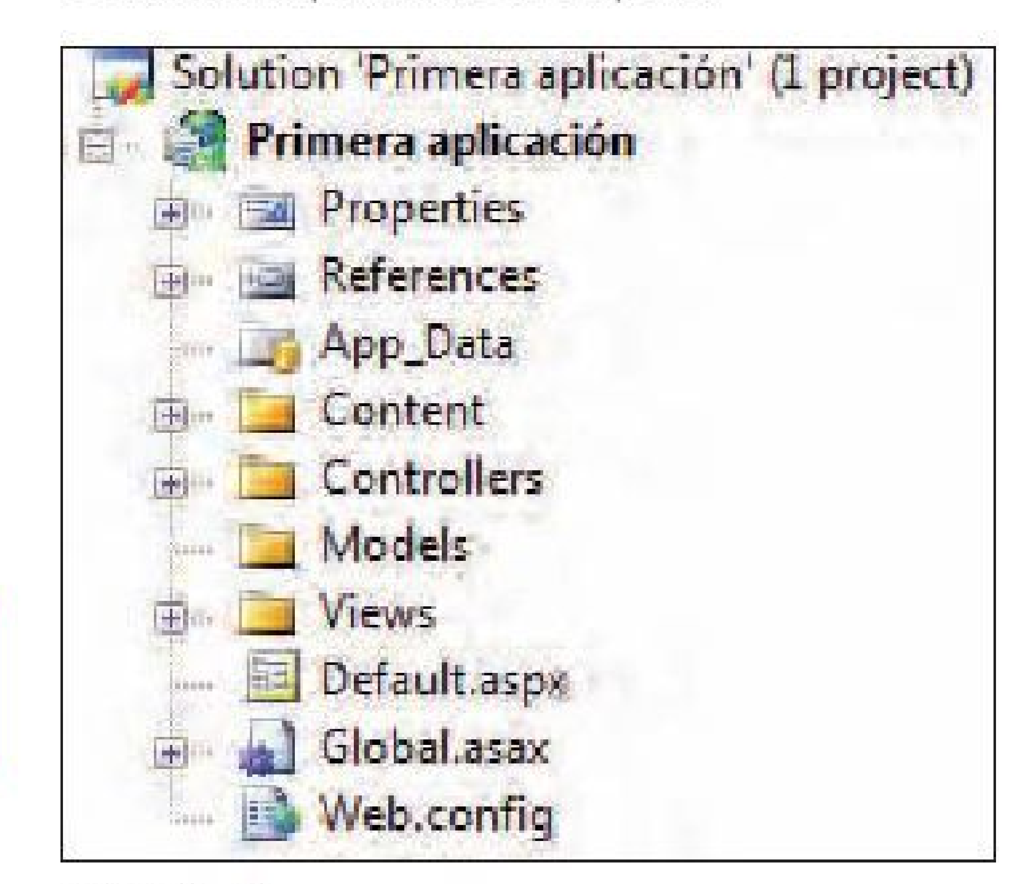


Figura 7.





Figura 8.

Recurso	Resultado final
http://localhost/Home	Obtiene el recurso "Home", el cual en este caso es una página de la interfaz gráfica.
http://localhost/Home/About	Obtiene el recurso "About", el cual en este caso es una página de la interfaz gráfica.

Tabla 1: ASP.NET MVC utiliza un estilo de organización de recursos llamado REST.

"En REST en vez de hacerse referencia a una página mediante su nombre, documento, servicio Web, etc., éste se organizan en una estructura lógica de sub-carpetas"

Esta es la primera diferencia importante con ASP.NET. El motivo de ello reside en que las aplicaciones MVC utilizan un estilo de arquitectura llamado REST (Represen-tational State Transfer o estado figurativo de transferencia), el que trata de organizar los recursos de una aplicación Web de forma "natural". En vez de hacerse referencia a una página mediante su nombre, documento, servicio Web, etc., ésta se organiza en una estructura lógica de sub-carpetas. Cada una de ellas representa un recurso en particular, el que podrá ser siempre obtenido si se escribe la dirección Web adecuada. La tabla 2 te muestra algunos ejemplos en la modalidad estándar y su contrapartida de estructura REST.

Como puedes apreciar, la página que lista los socios se llama ListaSocios.aspx en el mundo ASP.NET estándar, y será traída al explorador en el mundo REST mediante la URL http://servidor.com/Socios/Listar/. Básicamente cuando se indica este conjunto de carpetas, se usará un "controlador" llamado "Socios" en el servidor, el que sabe que /Lista/ se corresponde con la página ListaSocios.aspx.

```
/<Controlador>/<Acción>/
/Socios/Listar/
```

"La primera carpeta es el nombre del archivo de controlador, mientras que la segunda es la acción"

Socios representa aquí el nombre del controlador, el que es en realidad un archivo especial en el servidor, que contiene la relación entre las acciones (ej. Mostrar, Eliminar, Listar, etc.) y las páginas (en MVC se llaman vistas) u otros recursos (figura 9). Vamos a centrarnos ahora en la tercera fila de la tabla, la que emplea servidor.com/ Socios/ Mostrar/3/. Allí se usa un nuevo elemento, el que es en realidad un parámetro que será pasado al controlador.

```
/<Controlador>/<Acción>/<Parámetro>/
/Socios/Mostrar/3/
```

Es así que se estarán enviando dos datos al controlador llamado "Socios" (que como te dije antes, es un archivo especial en el servidor). El primero es la acción (la que identificará la página) y segundo el parámetro adicional. El controlador entonces sabrá que recurso obtener y retornar, así como también el valor de fila a leer en la base de datos.

Como puedes apreciar, muchos de los recursos en ASP.NET se relacionan 1 a 1 con la aproximación REST. No obstante, presta atención a la acción que modifica el socio... mientras que en ASP.NET usarías un solo formulario Web para representar las acciones de mostrar y modificar, aquí tendrás que dividir la tarea en 2 partes.

```
http://servidor.com/Socios/Mostrar/1/
http://servidor.com/Socios/Modificar/1/
```

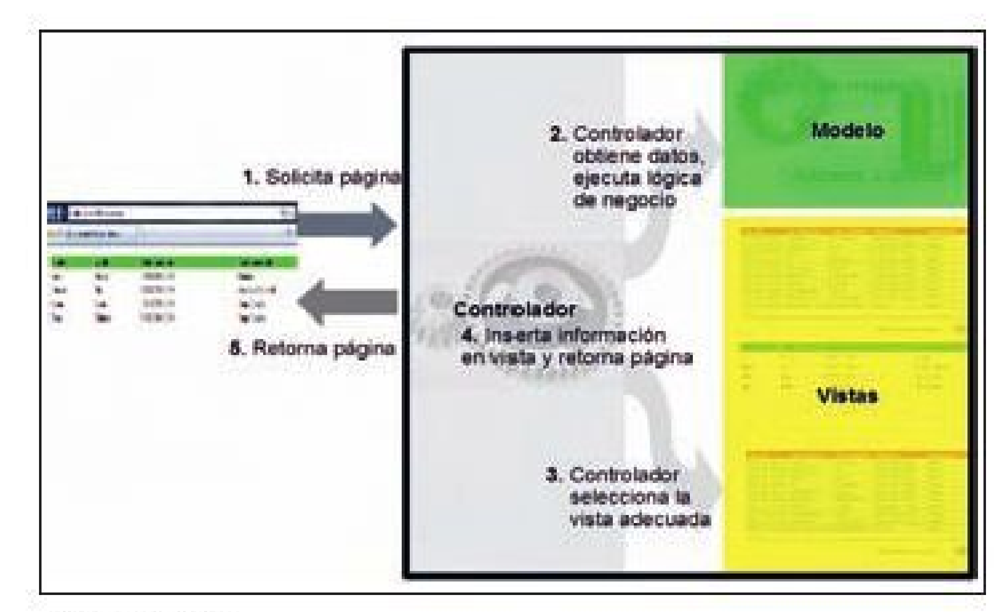


Figura 9.

La primera dirección traerá el formulario Web que muestra al socio pero cuando se presione el botón de modificar, se irá a "/Socios/Modificar/1/", en vez de reutilizarse la misma página (figura 10).

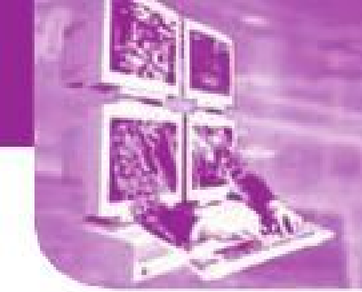
*Nota: En el próximo artículo veremos como pasar más de un parámetro.

El controlador de socios ejecutará la lógica asociada a la acción de modificar, y seguramente retornará otra página (o vista) con la lista de todos los miembros (/Socios/Lista/). Esto te asegura que no caerás en la trampa que vimos anteriormente, ya que contarás con distintos comportamientos en distintas direcciones. Esto último no quiere decir que una vista no pueda representar varios comportamientos, sino que en general las mismas se separan.

Como te puedes imaginar, REST no emplea el comando HTTP POST, el que es ampliamente usado para iniciar los eventos en ASP.NET. A diferencia, si se desea ejecutar una acción basta con el acceso la dirección Web específica (por ejemplo: http://servidor.com/Socios/Eliminar/13/). Esto también implica que las

Aproximación ASP.NET	Ahora en ASP.NET MVC (REST)	Descripción
http://servidor.com/PrincipalSocios.aspx	http://servidor.com/Socios/Principal/	Obtiene el recurso principal asignado al socio.
http://servidor.com/ListaSocios.aspx	http://servidor.com/Socios/Listar/	Lista todos los socios.
http://servidor.com/DetalleSocio.aspx&Id=3	http://servidor.com/Socios/Mostrar/3/	Obtiene los detalles del socio con identificador igual a 3.
http://servidor.com/ModificarSocio.aspx&Id=1	http://servidor.com/Socios/Mostrar/1/ http://servidor.com/Socios/Modificar/1/	Modifica el socio con el identificador 1.
http://servidor.com/EliminarSocio.aspx&Id=15	http://servidor.com/Socios/Mostrar/15/ http://servidor.com/Socios/Eliminar/15/	Elimina al socio con el identificador 15.
http://servidor.com/Socio22.doc	http://servidor.com/Socios/Detalle/Documentos/22/	Word relacionado al socio 22.

Tabla 2: Algunos ejemplos de organización de recursos mediante ASP.NET y su contrapartida en REST (estado figurativo de transferencia).



paginas no mantienen estado (o sesión), y que toda información deberá ser pasada en la línea de dirección URL.
Por último, se exhibe uno de los conceptos más importantes de REST, el que dice que todo recurso deberá ser expuesto de la misma forma, sin importar su tipo:

```
/Socios/Detalle/Documentos/22/
```

La línea anterior bajará un documento Word, aunque en el futuro podría ser cualquier otro tipo de elemento Web. Visto que se centra en direcciones sin nombre de archivos, la flexibilidad es mayor: todo está basado en la dirección URL del identificador del recurso, pero no en su tipo.

Archivos de Controlador en Microsoft ASP.NET MVC

Presta atención a la siguiente dirección, la que está expresada en formato REST.

```
http://<nombre del servidor>/socios/listar
```

Cada vez que se recibe este requerimiento en el servidor, se buscará dentro de la carpeta que contiene los controladores (aquí llamada "Controllers") para ver si existe un archivo llamado "SociosController.cs". De no contarse con el mismo entonces se exhibirá un error.

"MVC extrae el texto que se encuentra a continuación del nombre del servidor (Socios) y adiciona la palabra Controller.cs"

Lo que básicamente MVC hace es extraer el texto que se encuentra a continuación del nombre del servidor (Socios) y adicionar la palabra "Controller.cs". Para que todo funcione correctamente, este archivo tendrá que que heredar sus características de la clase "Controllers", la que es en definitiva quien brinda la infraestructura necesaria para que se intercepten las invocaciones Web. Veamos ahora un archivo controlador típico (listado 2).
Cada acción viene representada por un procedimiento en el archivo, el que es en realidad el responsable de gestionar una página en particular. Si por ejemplo, se indicase http://<nombre del servidor>/socios/ mostrar, entonces se buscaría y ejecutaría el procedimiento "mostrar" dentro del archivo "Socios Controllers.cs". Si por su parte fuese http://<nombre del servidor>/ empleados/ eliminar, entonces se buscaría la función "eliminar" dentro de "Empleados Controllers.cs"; tiene sentido ¿no? La instrucción RenderView es quien finalmente efectúa la magia. Para ello se debe incluir el nombre del archivo de vista a retornar, la que tiene que situarse dentro de una carpeta con el mismo nombre del controlador (pero sin la palabra "Controller"). (ver Figura 11)
Veremos en breve como escribir una página de vista, pero mientras tanto, me centraré en la forma de pasar parámetros.



Figura 10.

Controladores y parámetros

Usaré un ejemplo para explicar como funcionan los parámetros. Lo primero que deberás hacer entonces es adicionar dentro de la carpeta del modelo (la que como te comenté anteriormente, es donde se almacena todo lo relacionado a datos y lógica) un nuevo archivo de tipo Linq a clases SQL. En este caso lo llamaré clubDeportivo y contendrá las diferentes clases que representan tablas de una base de datos (puedes bajar todos los ejemplos del sitio Web de la revista). (ver Figura 12) Una vez hecho esto, tan solo tendrás que arrastrar las tablas del explorador de servidores al diseñador del archivo de Microsoft Linq a SQL; tal como exhibe la figura 13.
Imagínate que deseas tener una página de vista que sea capaz de mostrar todos los socios o uno en particular. Para ello, lo primero que deberías pensar es en como conformar la dirección:

```
http://<nombre del servidor>/socios/
listar/30/
```

El próximo paso consiste en agregar un parámetro al procedimiento de la clase controladora, el que le dirá a ASP.NET MVC que quieres leer el valor (listado 3):

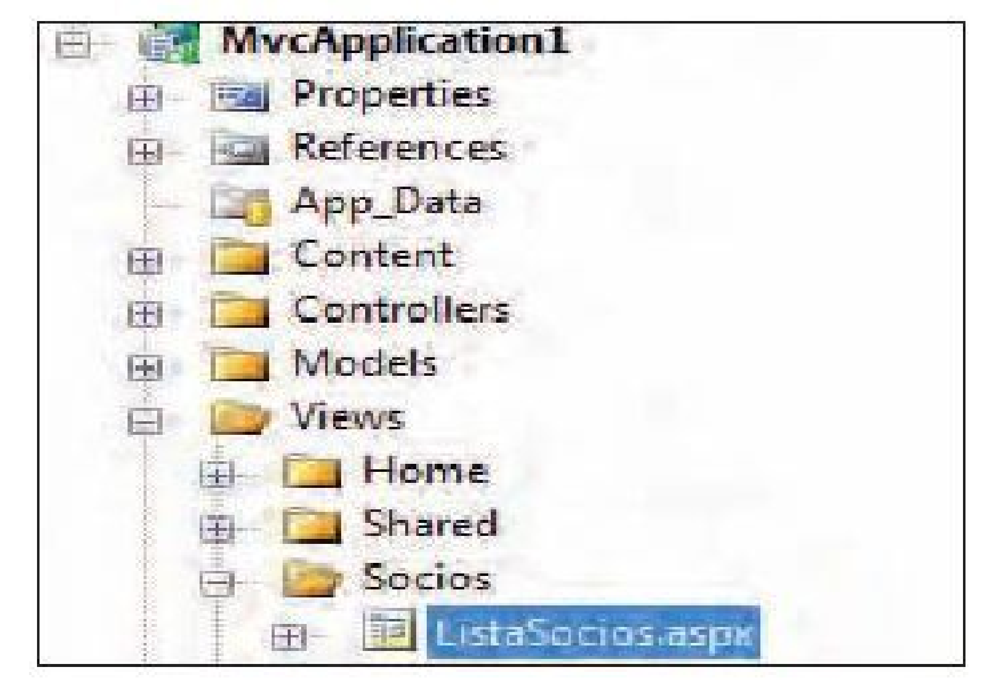


Figura 11.

"En MVC se pueden emplear los tipos que soportan nulos, para así indicar que el mismo podría no estar presente"

Se define un entero con el símbolo de interrogación al final, para indicar que la variable podría contener un valor nulo. Veamos entonces la implementación del método listar (listado 4):
Hay dos cosas importantes en el ejemplo, la primera es que se verifica el valor de "numsocio" para saber si se toma uno u otro camino. La segunda (¡y mas importante!) es que el conjunto de información obtenido de la base de datos se adiciona como segundo parámetro del método RenderView. Esto hará que la página de vista reciba la información y posteriormente la pueda gestionar y mostrar (veremos en breve como se lleva esto adelante).

"RenderView permite especificar el nombre de la vista, así como también información a pasar a la misma"

Existe un pequeño paso más que deberás efectuar en ASP.NET, para decirle a MVC que efectivamente se quiere pasar el parámetro al controlador. Este último paso enlazará finalmente el valor especificado en la dirección Web con la función de la clase. Deberás entonces ir el archivo de configuración del proyecto "Global.asax" y escribir en la función RegisterRouter las líneas que se encuentran resaltadas al comienzo del procedimiento (listado 5).
Lo que parece chino mandarín es simplemente una forma de indicarle a ASP.NET MVC que hay

LISTADO 2

```
public class SociosController : Controller
    {
        public void Listar()
        {

            RenderView("ListaSocios");
        }

    }
```

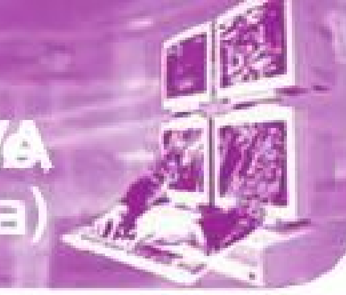




Figura 12.

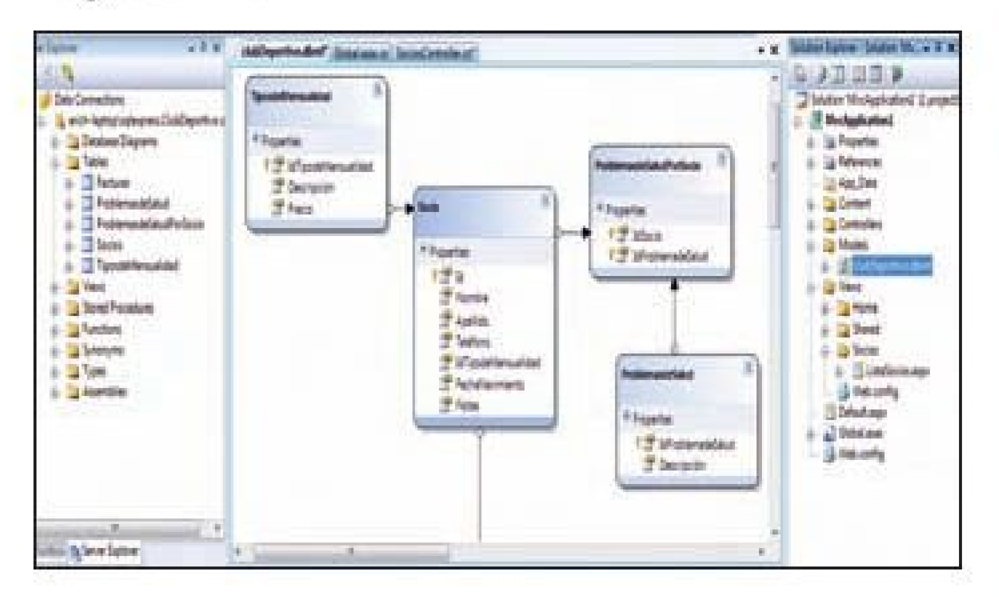


Figura 13.

LISTADO 3

```
namespace MvcApplication1.Controllers
{

        public void Lista(int? numSocio)
        {
        }
}
```

LISTADO 4

```
            if (numSocio != null) {

                vd.Socios = (from s in cd.Socios
                             where s.Id ==  numSocio
                             orderby s.Nombre
                             select s);
            }else{
                vd.Socios = (from s in cd.Socios
                             orderby s.Nombre
                             select s);
            }

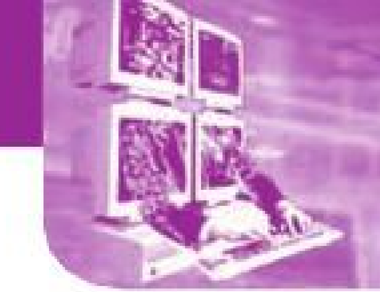
            RenderView("ListaSocios",vd);
```

**Nota de listado 4: la variable vd es una pequeña clase que he diseñado, la que contiene una propiedad Socios que almacena las filas.*

un nuevo parámetro llamado "numsocio", el que deberá ser pasado a la clase controladora "Socios". Basta con copiar el texto descrito para que todo funcione correctamente. Si en el futuro tienes otra página con parámetros, entonces recuerda adicionar estas líneas y reemplazar el valor de "Controller" y "Socios" por el nombre del nuevo controlador y acción. A su vez, podrías tener diferentes acciones dentro del controlador, las que empleasen la misma lógica de negocio pero diferentes vistas. Esto daría una flexibilidad mayor al ofrecer distintas presentaciones visuales para una misma tarea.

Las vistas en Microsoft ASP.NET MVC

Como comenté anteriormente, las vistas son archivos "especiales" que representan la interfaz gráfica. Ellas no tienen en general acceso direc-



to al modelo y es el controlador el que la invoca e inyecta la información a desplegar. A diferencia con ASP.NET, los archivos ganan sus cualidades de una clase llamada ViewPage; el listado 6 exhibe un pequeño ejemplo del código de detrás de escena.

También se cuenta con otro archivo que representa la interfaz gráfica, la que funciona de forma relativamente similar a un formulario Web. Como podrás ver en el listado 7, he creado una tabla estándar HTML que muestra los distintos socios de un centro deportivo. Cada una de las filas son creadas dinámicamente mediante la utilización de la estructura "foreach", quien efectúa un bucle por los diferentes elementos de la variable "viewdata".

Pero... ¿De donde sale ViewData?

ViewData es en realidad un objeto implícito que funciona como una bolsa conteniendo la información indicada en el controlador, como segundo parámetro del método RenderView (por ejemplo: RenderView ("ListaSocios", vd)). Cualquier variable allí especificada se cargará dentro de ViewData. No obstante, obtendrás un error si intentas compilar el código del listado 7. El motivo es que si bien ViewData contiene las filas de los distintos socios, ellas se exponen como un objeto del tipo "Object", en vez de mantenerse el tipo original.

Es posible entonces informarle al compilador el tipo de datos guardado en ViewData, con el fin de que los métodos y propiedades de la clase original estén disponibles al escribir la interfaz gráfica.

Esto traerá como consecuencia de que se ofrezcan los miembros mediante intellisense, así como también la validación de nombres de métodos, propiedades y tipos de dato. Para lograr esto hay que llevar adelante dos pasos:

- Importar el espacio de nombres donde se encuentra la definición de la clase contenida en viewData (ella está dentro del espacio de nombres del controlador (Mvc Application1. Controllers), aunque podría situarse en cualquier otro lado).
- Escribir el nombre del tipo de datos específico contenido en ViewData, que en este caso es "SociosControllerViewData" (así es como denominé la clase y es lo que se indicó como segundo parámetro en RenderView)

Sé que la sintaxis es bastante extraña (especialmente la forma de indicar la clase), pero es la vía en que el compilador lo espera. (ver Listado 8)

Esto hará que los métodos y propiedades del objeto contenido dentro de ViewData se desplieguen automáticamente al escribirla

LISTADO 5

```
    public class GlobalApplication : System.Web.HttpApplication
    {
        public static void RegisterRoutes(RouteCollection routes)
        {

            routes.Add(new Route("{controller}/{action}/{numSocio}", new
MvcRouteHandler())
            {
                Defaults = new RouteValueDictionary(new { controller =
"Socios", action = "ListaSocios", id = (int?)null }),
            });
        }
```

LISTADO 6

```
namespace MvcApplication1.Views.Socios
{
    public partial class Lista : ViewPage
    {
    }
}
```

LISTADO 7

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="ListaSocios.aspx.cs"
Inherits="MvcApplication1.Views.Socios.Lista" %>

 <table style="width:100%;">
    <tr style="background-color:Lime">
        <td>
            <b>Nombre</b></td>
        <td>
            <b>Apellido</b></td>
        <td>
            <b>Fecha de nacimiento</b></td>
            <td>
            <b>Tipo de mensualidad</b></td>

    </tr>
        <%
            foreach (var c in ViewData.Socios)
          { %>

    <tr>

        <td><%=c.Nombre %></td>
        <td><%=c.Apellido %></td>
        <td><%=c.FechaNacimiento %></td>
        <td><%=c.TiposdeMensualidad.Descripción %></td>
    </tr>
        <%} %>
</table>
```

LISTADO 8

```
using MvcApplication1.Controllers;

namespace MvcApplication1.Views.Socios
{
    public partial class Lista : ViewPage<SociosControllerViewData>
    {
    }
}
```

**Nota de listado: Indicando el tipo de datos que se encuentra dentro de ViewData.*

interfaz gráfica de la vista, así como también se realicen las validaciones respectivas. El resultado final será entonces el de la figura 3.

Hasta aquí he llegado con este artículo sobre Microsoft ASP.NET MVC. El próximo mes me encargaré de mostrare algunas características avanzadas de la nueva plataforma, para que puedas seguir incursionando en el tema. Mientras tanto, espero que hayas disfrutado del mismo y que emplees los restantes días del mes para bajar, instalar y probar el paradigma de modelo, vista y controlador. Como siempre, espero tus comentarios a través de tuscomentariosmeimportan@ hotmail.com SP

Participa en JavaCup 2008, el torneo de fútbol virtual Java



El ganador de la JavaCup se anunciará durante la OpenJavaDay 2008, que se celebrará el 26 y 27 de junio

Organizan







Patrocinan





Idea original y desarrollo del framework
JORGE RUBIRA

JavaCup 2008

Es un concurso que consiste en un torneo de fútbol virtual, basado en eliminatorias, donde cada equipo será una clase Java que implementará la estrategia del mismo, apoyándose en un framework creado para tal efecto. Para participar, sólo tienes que implementar tu equipo (una clase Java) y enviárnoslo.

Premios:
- 1500 euros para el 1º - 1000 euros para el 2º
- 500 euros para el 3º - 250 euros para el 4º

además de una suscripción anual a la revista *Sólo Programadores*.

http://javacup.javahispano.org





Desarrollo de una aplicación Java distribuida

IGNACIO VERDÚ GALIANA - Consultor Técnico de Software

J2EE proporciona un modelo de desarrollo software basado en componentes que pueden estar distribuidos en diferentes servidores. Vamos a estudiar que significa esto y que posibilidades de desarrollo nos permite.

Introducción

Las tecnologías de la programación, ofrecen una serie de posibilidades para el desarrollo e implementación de sistemas informáticos que permiten ofrecer soluciones a unos requerimientos dados. No vamos a discutir aquí ninguna metodología de desarrollo en concreto (*Proceso Unificado de Rational, Métrica, etc.*.), pero si que necesitamos, por la explicación en si, mostrar el sistema que queremos conseguir y cómo queremos que funcione.

Es importante conocer de una manera global, de qué herramientas disponemos y cómo podemos interconectarlas para conseguir que todo funcione de una manera sincronizada. Cuando desarrollamos un sistema, es posible determinar partes críticas susceptibles de una mayor demanda de capacidad de procesamiento, hecho que nos permite ya no solo diseñar, sino más bien prever la instalación y los requerimientos tanto software como hardware que vayamos a necesitar.

Normalmente, una de las etapas iniciales en el desarrollo de un proyecto es, o debería ser, la definición de la plataforma o plataformas software que se van a utilizar para el desarrollo del mismo. Además, se suele determinar qué subproyectos o qué aplicaciones van a ser necesarias para conseguir el objetivo final; esto permite entre una serie de ventajas: modularizar el desarrollo, subcontratar determinadas partes, y, en general disponer de una visión global de las diferentes necesidades concretas del proyecto.

Este es un artículo sobre desarrollo en *J2EE*. En el mismo, vamos a estudiar de una manera general una serie de posibilidades dentro del amplio abanico de las ofrecidas por la plataforma *JAVA* para el desarrollo de una aplicación. Eso si, no va a ser una aplicación web al uso, sino que definiremos un modelo de negocio que queremos implementar para el cual, nos harán falta dos aplicaciones según las especificaciones; estas dos aplicaciones además, necesitarán disponer de un canal de comunicación pero que las acople lo mínimo necesario.

La labor de un arquitecto software, consiste entre otras cosas en definir a nivel conceptual qué necesidades de desarrollo presenta un proyecto y para ello, creo que el lector estará de acuerdo conmigo, es necesario que conozca las posibilidades que ofrece una tecnología, repito, en nuestro caso *J2EE*. Todo esto es lo que hace que tenga sentido la planificación de proyectos, los plazos de entrega que seguro que muchos lectores han sufrido y la gestión entre diferentes equipos de desarrollo trabajando en un mismo proyecto.

Presentación del sistema a desarrollar

Nos han encargado el desarrollo de una aplicación Terminal Punto de Venta (TPV) para una red de kioscos de prensa. La empresa dispone de una central de reparto/reservas desde la que se envían las publicaciones diariamente a las diferentes tiendas de la cadena. Con el TPV, pretende establecerse un sistema para que el kiosco pueda gestionar de una forma sencilla las reservas de publicaciones que hacen sus respectivos clientes. Se quiere dividir el desarrollo en dos sistemas distintos, por un lado la central de reservas y por otro, la red de kioscos.

Según las especificaciones podemos comprobar que necesitamos desarrollar dos aplicaciones distintas que se puedan comunicar, pero que, dependan lo mínimo la una de la otra, queda sobreentendido (no me hagáis caso los que trabajáis en esto, no sobreentendáis nunca un proyecto real), que las aplicaciones han de estar perfectamente sincronizadas para que se puedan cumplir los objetivos de negocio; estas dos aplicaciones, además, no tienen por qué estar ejecutándose sobre un mismo servidor, por lo que tendremos que ver que la comunicación sea lo más flexible posible.



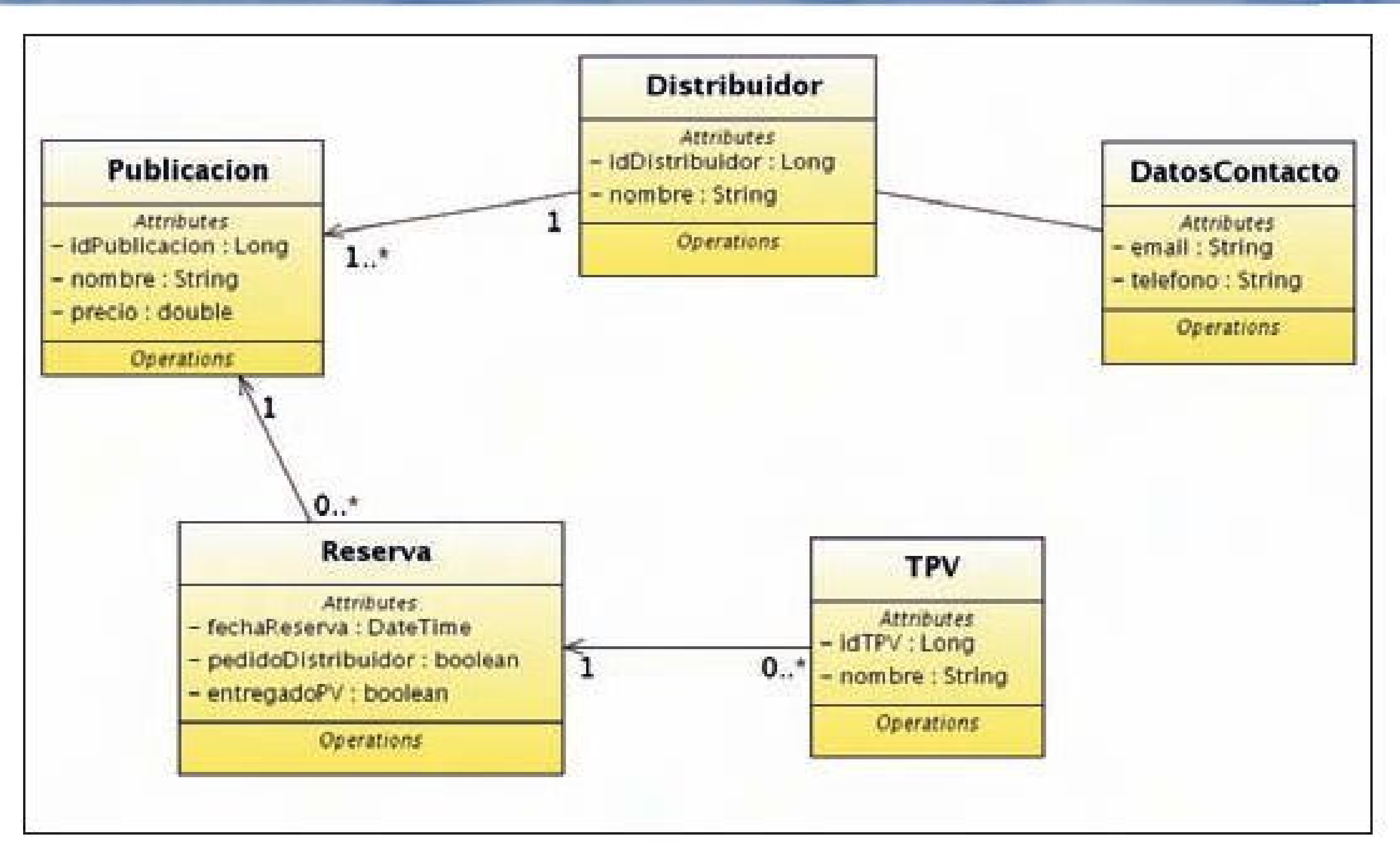


Figura 1. Modelo Central de Reservas.

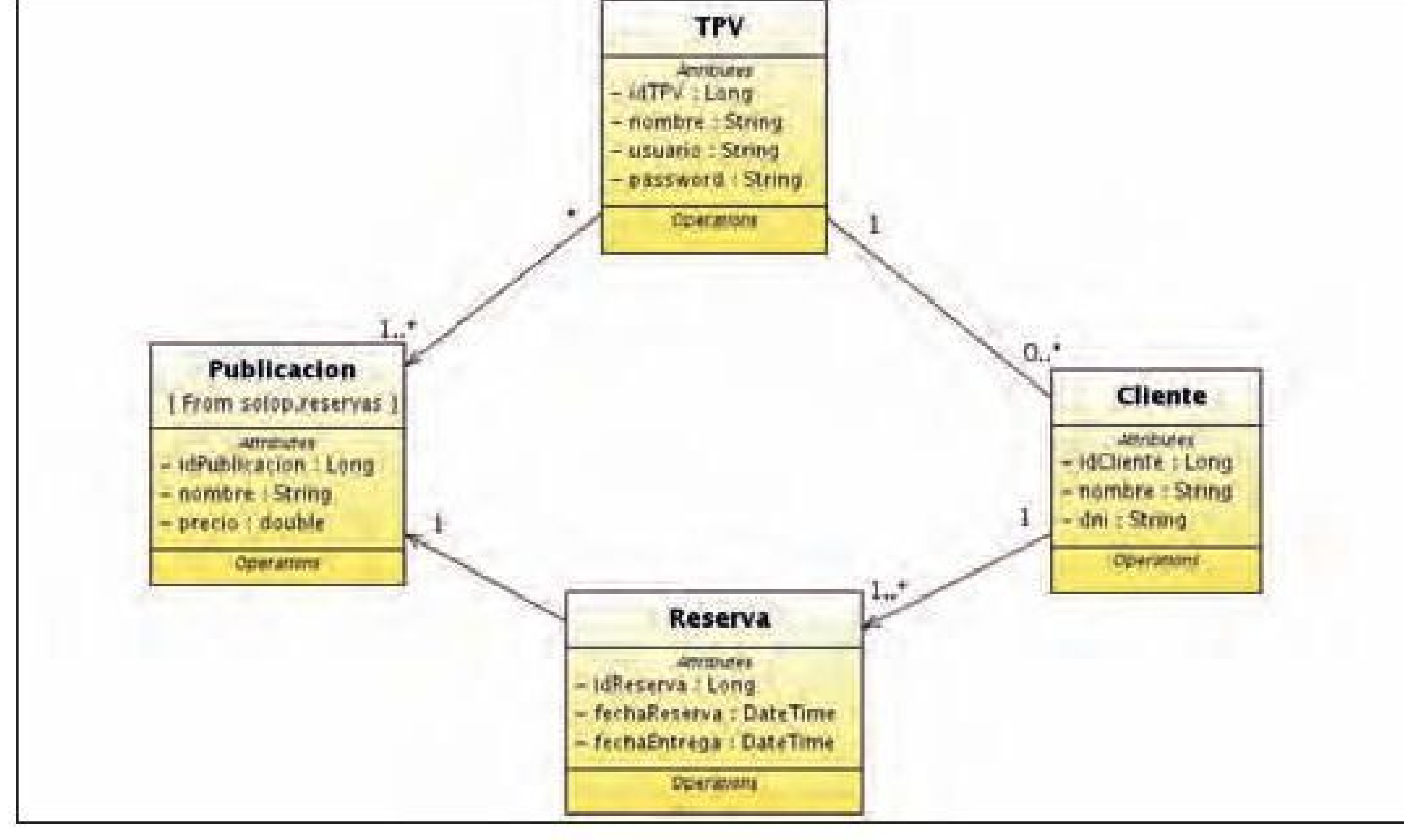


Figura 2. Modelo Terminal Punto de Venta.

A primera vista, un programador puede pensar que todo esto se soluciona con una simple aplicación web; aunque esto es cierto, nosotros vamos a introducir un mayor grado de complicación porque, esto a la larga nos va a permitir ver el desarrollo de aplicaciones de una forma más global en un contexto mucho más amplio cubriendo mayores necesidades y escenarios posibles. Eso si, esto es un tutorial de ejemplo y las aplicaciones son de por si muy sencillas, por no decir triviales.

Después de darle vueltas al problema que se nos presenta, como arquitectos software, hemos adoptado la decisión de diseñar e implementar dos sistemas: uno para la Central de Reservas o CR y otro para el Terminal Punto de Venta o TPV.

Por supuesto y como buenos programadores, a pesar de que nos suele gustar empezar a escribir código en seguida, vale la pena que nos pongamos un rato el traje de analista y nos pongamos a pensar en los modelos de negocio que queremos implementar para cada una de las dos aplicaciones. Fijaros en que para la definición de un modelo, estamos todavía en una fase de diseño abstracta en la que todavía no estamos hablando siquiera de qué lenguaje de programación vamos a utilizar.

La Central de reservas es un sistema encargado de gestionar los pedidos que desde los diferentes terminales puedan hacerse, no necesita saber datos de los clientes de dichos terminales, simplemente los datos del terminal en cuestión y la publicación reservada. (ver Figura 1).

El TPV, va a ser una aplicación, disponible en cada uno de los kioscos de la red (accesible mediante usuario/password) a través de la cual el vendedor de dicho kiosco podrá gestionar de una forma muy rápida las posibles reservas de publicaciones que hagan sus clientes. (ver Figura 2).

Acabamos de describir de una manera somera los modelos de las dos aplicaciones que hablábamos al principio. Nos queda otro paso, insisto, fíjese el lector en que todavía no hemos programado nada, el caso es que ahora nos falta describir como queremos implementar las aplicaciones y como van a estar ejecutándose. Nos falta definir la arquitectura de desarrollo.

Arquitectura de desarrollo

El modelo, desde luego es importante puesto que define una visión conceptual de lo que queremos conseguir y sirve además, como sabréis, como manera para comunicarse entre personal informático y personal usuario (lo contrario del informático); hoy en día, de hecho, existe una corriente de metodologías en las que se empieza el desarrollo con un modelo (MDA Model-driven-Architecture) y a partir del mismo se generan "artefactos" software que sirven de base para la programación posterior del sistema; como veremos en apartados a lo largo de este artículo, nosotros seguiremos una política de desarrollo en esta línea mediante el uso de diversas técnicas, entre las que destacamos los modelos de "POJO" del inglés "Plain Old Java Object" y de las anotaciones en *JAVA*.

En cualquier caso, antes de empezar a programar y, puesto que somos "expertos" en desarrollo software, vamos a presentar un plan de implementación en el que propondremos los componentes o soluciones software a desarrollar y las necesidades hardware necesarias para soportar los mismos. Básicamente y como hemos mencionado ya, dispondremos de dos aplicaciones, cuyo interfaz será web en ambos casos y cada una estará implantada en un servidor distinto; por las necesidades del proyecto, estas aplicaciones tendrán que estar comunicadas; nosotros aplicaremos un diseño en el que estarán acopladas lo mínimo posible,

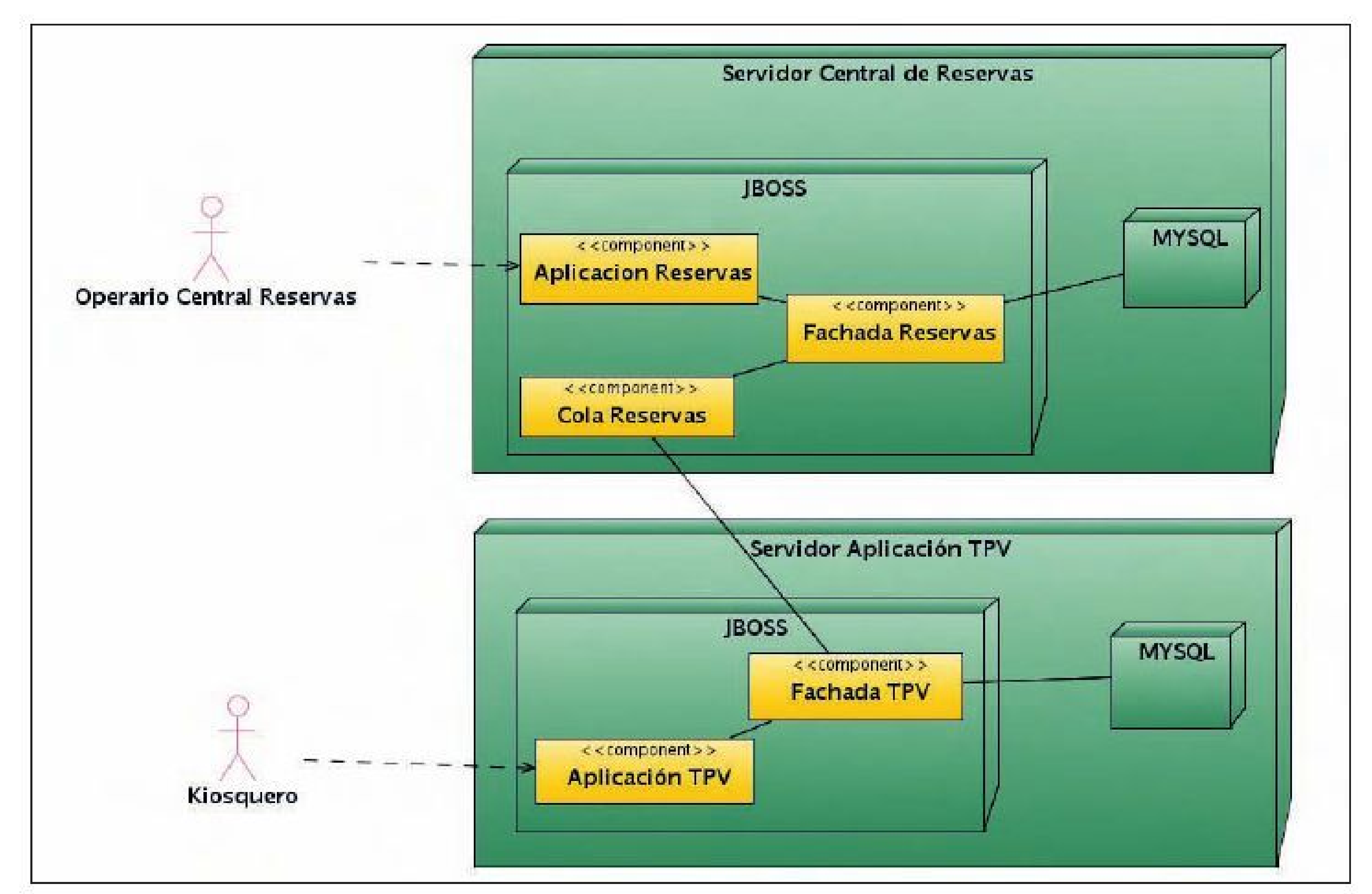


Figura 3. Implantación desarrollo.



mediante un protocolo de comunicaciones que también expondremos más adelante. (ver Figura 3).

Aquí conviene recapitular y refrescar unos conceptos muy importantes y que son ampliamente utilizados hoy en día en los proyectos de desarrollo software profesional y que además son conceptos que pueden aplicarse en diferentes tecnologías, no solo en *JAVA*.

Modelo Vista Controlador (Struts)

Este es hoy en día uno de los patrones de diseño más conocidos y más ampliamente utilizados en proyectos. Consiste, en que a la hora de organizar el código de una aplicación, este se subdivide en tres módulos donde se pretende, se consigue si se hace bien, tener dos módulos que son independientes entre si: el modelo y la vista o interfaz, los cuales, son coordinados a través del controlador. Para entendernos y por si la explicación resulta algo rebuscada, consiste en que por ejemplo si estamos desarrollando una aplicación web, las páginas jsp, contendrán sólo código que mostrará datos y no contendrán en ningún caso código para el acceso y manipulación de los mismos, esto se lo dejaremos al otro módulo, al que hemos llamado modelo.

¿Todo esto para que sirve?. Pues existen multitud de ventajas, una de ellas por ejemplo es la reutilización de código; otra sería que podríamos cambiar de interfaz de una manera más sencilla, puesto que el código de acceso no está acoplado a la misma. En el mundo *Java* existen varios *"frameworks"* de desarrollo que siguen el MVC, nosotros

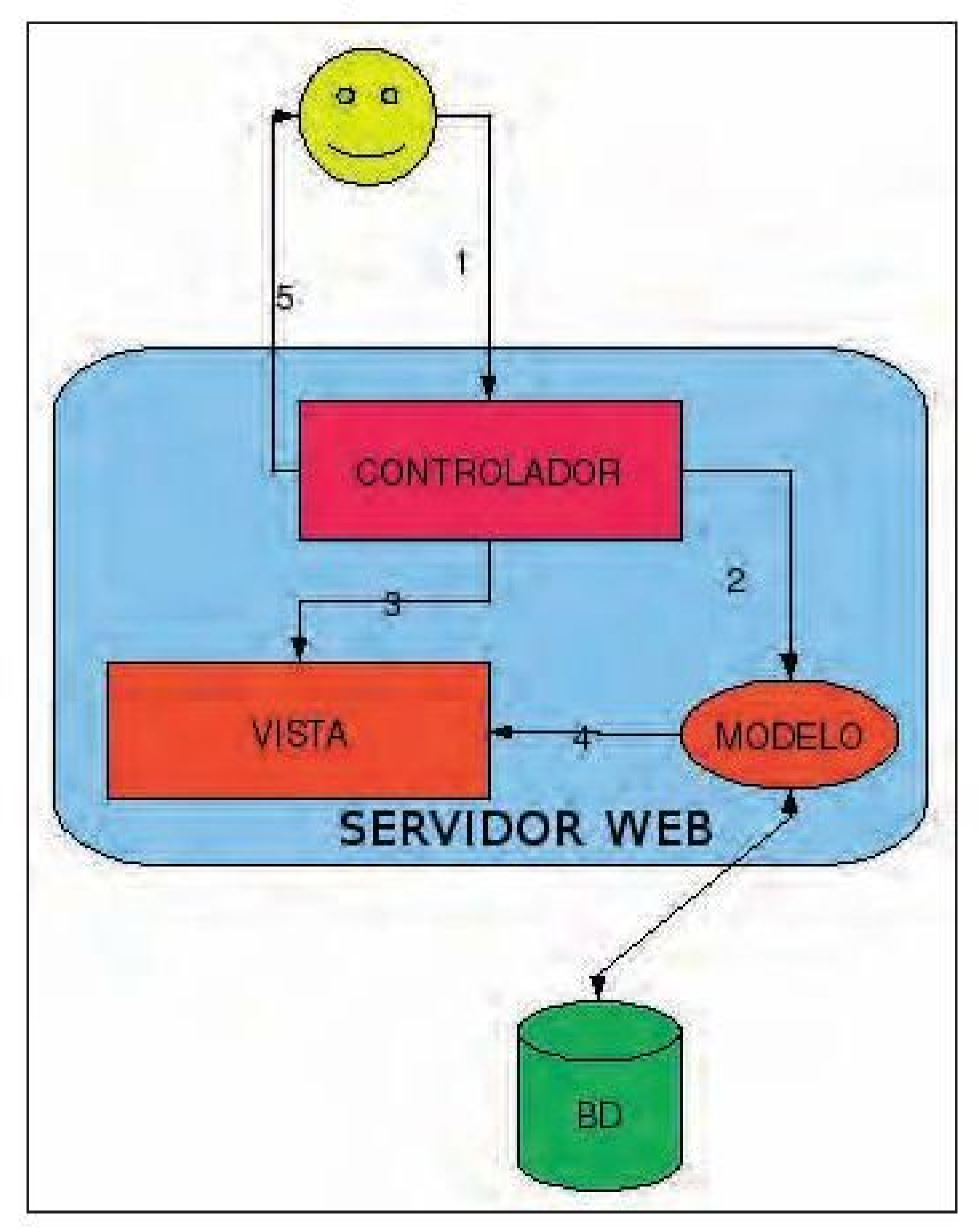


Figura 4. Modelo 2, Struts.

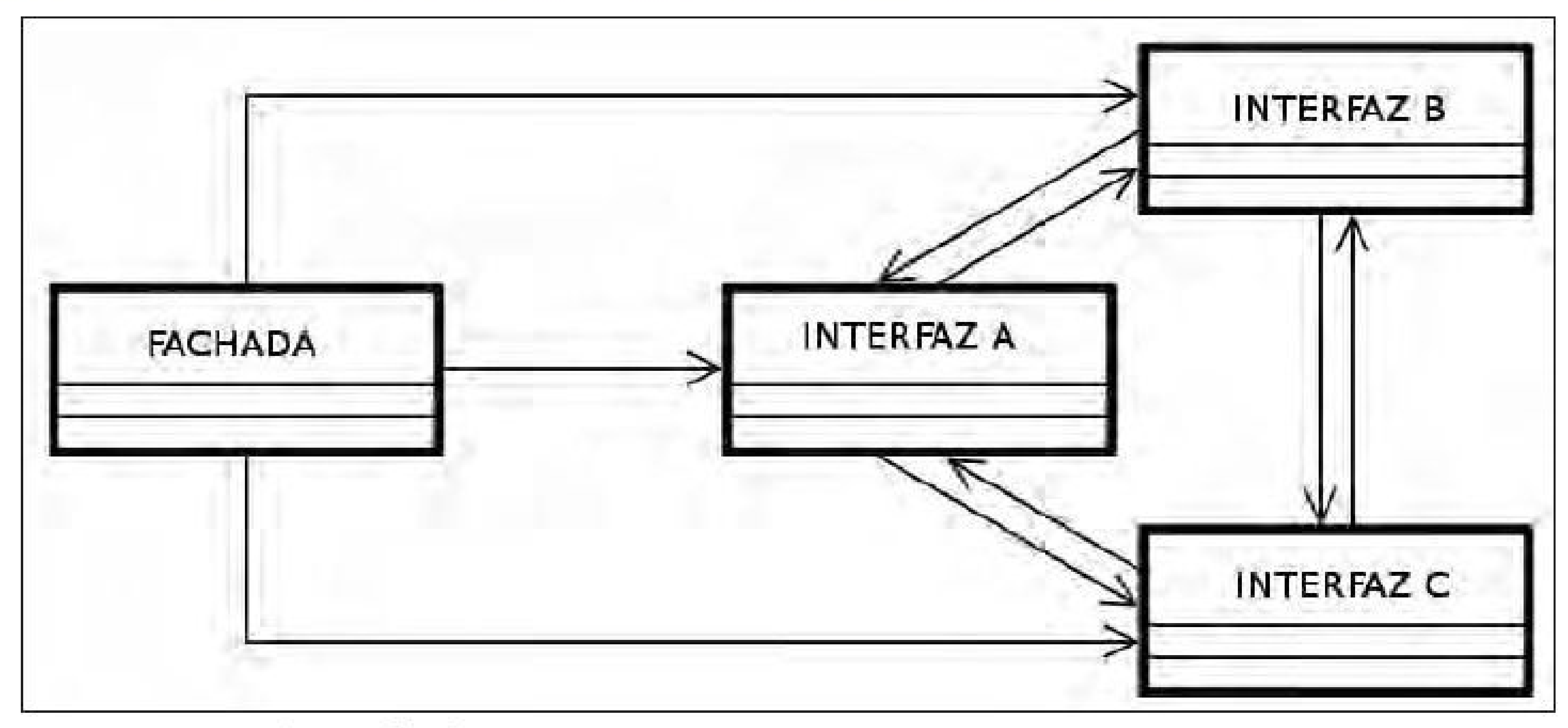


Figura 5. Patrón Fachada.

en esta serie de artículos vamos a utilizar uno que está ampliamente extendido: *Struts*. (ver Figura 4).

Cabe anotar que *Struts* básicamente es un "controlador", es lo que nos va a permitir "separar" pero tener unidos el modelo y la vista. Esto lo consigue definiendo lo que llama "Actions" o "Acciones", que son los puntos en el código donde *Struts* intercepta y da respuesta a las peticiones hechas por el usuario de la aplicación web. En cada uno de estos "Actions" *Struts* quizá necesite acceder a la base de datos, pero lo hará a través del modelo que habremos definido e implementado en otra parte.

Dentro del *Struts*, podemos destacar una serie de "plugins" que hacen que la vida del programador sea mucho más cómoda y permiten por lo tanto, más tiempo para tomar cafés; me refiero en concreto a dos de ellos: "Validator" y "Tiles".

"Validator", permite definir validaciones de formularios de forma declarativa y fuera del código de las páginas jsp; estas validaciones además pueden implantarse tanto a nivel de cliente, es decir con *javascript* en el navegador, como a nivel de servidor; bueno a nivel de servidor siempre han de definirse claro está. Básicamente "Validator" hace que los datos que llegan a los "Action", estén validados a nivel de formato, como por ejemplo, si una cadena ha de tener X caracteres, si han de seguir una expresión regular como podría ser un código postal o un teléfono, etc.

"Tiles", dicho de forma simple , permite definir herencia de objetos a nivel de *jsps*. A la hora de la práctica, "Tiles", nos sirve para definir "vistas" o páginas *jsp* genéricas, donde se define el esqueleto de las diferentes páginas de la aplicación y donde se dejan "marcas de contenido" donde van los datos producidos por el modelo desde los "Action". Toda la aplicación web estará formada por "instancias" de esos esqueletos en los que se muestra el contenido, pero separando el mismo del diseño individualizado por página. Esto nos permitirá de una forma sencilla su posible cambio o reutilización de una forma además más rápida que sin la utilización del mismo.

Seguridad Aplicaciones Web

Un aspecto primordial es tener en cuenta la seguridad de las aplicaciones y de los componentes que vayamos a desarrollar; pensemos en dos aspectos: privacidad del código y seguridad de las acciones de los usuarios de los programas.

Para proteger el código, no queremos que nadie sin nuestro permiso investigue lo listos que somos, disponemos de un mecanismo previsto en la especificación de aplicaciones web en *JAVA* que consiste en ubicar el mismo, por debajo del directorio "WEB-INF". De esta forma es sabido que nadie podrá editar las páginas jsp ni ficheros de configuración ni demás artefactos software.

En cuanto a la seguridad o acceso a las acciones de nuestra aplicación, disponemos básicamente de dos mecanismos contrapuestos: seguridad programada y seguridad declarativa. La seguridad programada consiste en que en el código de cada función ejecutamos un test para saber si el usuario que está en ese momento accediendo al mismo tiene permisos suficientes para hacerlo; a la hora de la verdad en vez de escribir el mismo código muchas veces lo que hacemos en *JAVA* es implementar un *"Filter"*. El otro sistema es la seguridad declarativa que, al contrario que la programada hace que un determinado código sea agnóstico en cuanto al acceso.



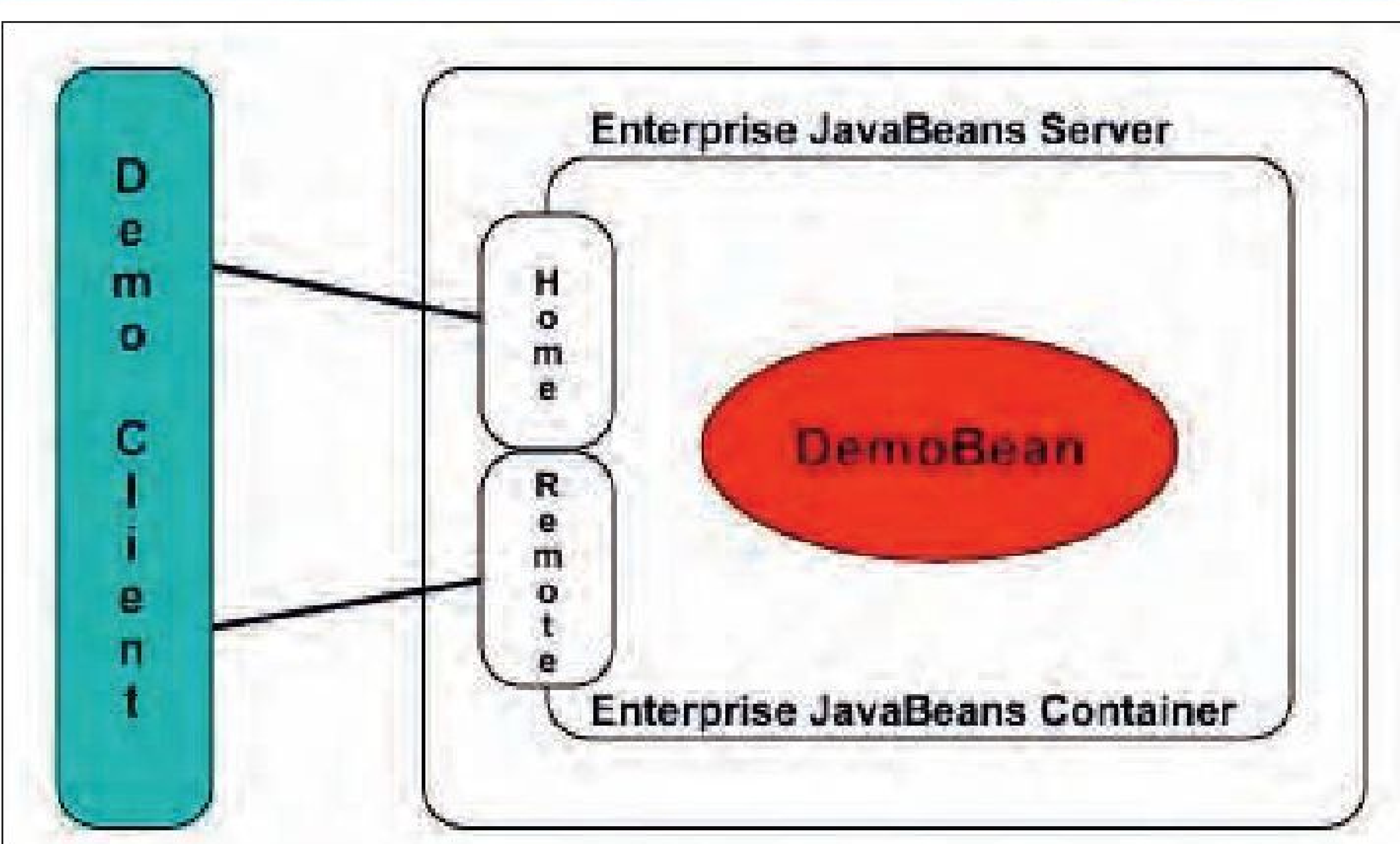


Figura 6. Llamadas EJB Sesión.

Modelo de Negocio (EJB3 y EJB Sesion sin estado)

En el apartado anterior hemos hablado del "controlador" y de la "vista" en una aplicación web desarrollada en *Struts*, nos falta saber cómo vamos a acceder a los datos. Para ello a su vez, vamos a diferenciar dos aspectos del modelo: por un lado tendremos que poder crear, modificar y consultar datos, es lo que llamamos "Capa de Persistencia", nosotros en nuestro caso vamos a implementar la misma con uno de los servicios que nos ofrece *JAVA* mediante *EJB3;* por otro lado para acceder a dichos *EJB3*, lo haremos mediante la utilización de otro conocido patrón de desarrollo, el patrón "Facade" o en castellano "Fachada".

¿El patrón "Fachada", para que sirve?. Según la *WikiPedia*: "sirve para proveer de una interfaz unificada sencilla que haga de intermediaria entre un cliente y una interfaz o grupo de interfaces más complejas." Lo que queremos es que nuestra aplicación no dependa de una implementación concreta de la "Capa de Persistencia"*;* lo que hacemos es definir un interfaz con las operaciones de acceso a datos que vayamos a necesitar, e implementamos dichas operaciones mediante *EJB3* tal y como hemos mencionado. De esta forma podríamos cambiar el acceso a la base de datos sin tocar el código que accede a los mismos, cambiando la implementación de dicho interfaz. Como el lector sabrá, uno de los principios básicos de la programación orientada a objetos, es el "programar en base al interfaz, no a la implementación concreta del mismo", la ventaja acabamos de mencionarla al decir lo de poder cambiar implementaciones sin tocar el código. (ver Figura 5).

En *Java*, para el desarrollo de la "Fachada", nos vienen como hechos a medida los *EJB*s de sesión sin estado. Además, estos componentes tienen una ventaja y es que permiten definir dos tipos de interfaces, una local y otra remota; esto quiere decir, que podremos acceder a la fachada bien de forma que esté instalada en el mismo servidor que el programa que la llama (Fachada Local) o en un servidor remoto (Fachada Remota). El lector puede ir imaginándose como aumentan las posibilidades de implantación de sistemas distribuidos basados en tecnología Java. (ver Figura 6).

Comunicación desacoplada (Colas mediante MDB)

Nos queda un último elemento a explicar del planteamiento inicial de desarrollo que hemos hecho, este es la comunicación entre el TPV y la Central de Reservas. Vamos a partir de la base que según nuestro análisis inicial del sistema, no es necesario que en el momento en el que un cliente y por lo tanto cuando desde el TPV se haga una reserva, saber en qué momento concreto tendrá que recogerla, sino que siempre se le dirá que pase a recogerla otro día. No se si esto es factible en el mundo real o no, permítame el lector esta licencia para que pueda continuar con el ejemplo que tengo en mente. Básicamente lo que sucede es que el TPV no necesita esperar una respuesta inmediata de la Central de Reservas, su comunicación puede y va a ser por lo tanto asíncrona.

¿Comunicación asíncrona, eso que es?. Es conveniente que un desarrollador comprenda qué significa y qué diferencia hay entre llamadas remotas síncronas/asíncronas y qué artefactos o componentes software existen en *Java* para su utilización e implementación. Una llamada de un proceso a otro es síncrona, cuando el que llama se queda en estado de espera hasta que recibe una respuesta; una llamada es asíncrona cuando el proceso que llama no se queda en espera de respuesta sino que continúa su línea de procesamiento. El ejemplo clásico de protocolos síncronos/asíncronos es la comunicación mediante conversación telefónica que sería un ejemplo de llamada síncrona o, el de envío de emails que representa a la comunicación asíncrona. (ver Figura 7).

En *Java*, las llamadas a *EJBs de sesión* son síncronas, por ejemplo cuando desde una "Action" de *Struts* llamemos a la "Fachada", el proceso del "Action" se queda en espera hasta que la llamada finaliza; evidentemente este tiempo es corto, pero no es ese el sentido argumental que estamos siguiendo ahora. En cambio, por ejemplo, y viendo nuestras especificaciones de negocio, el

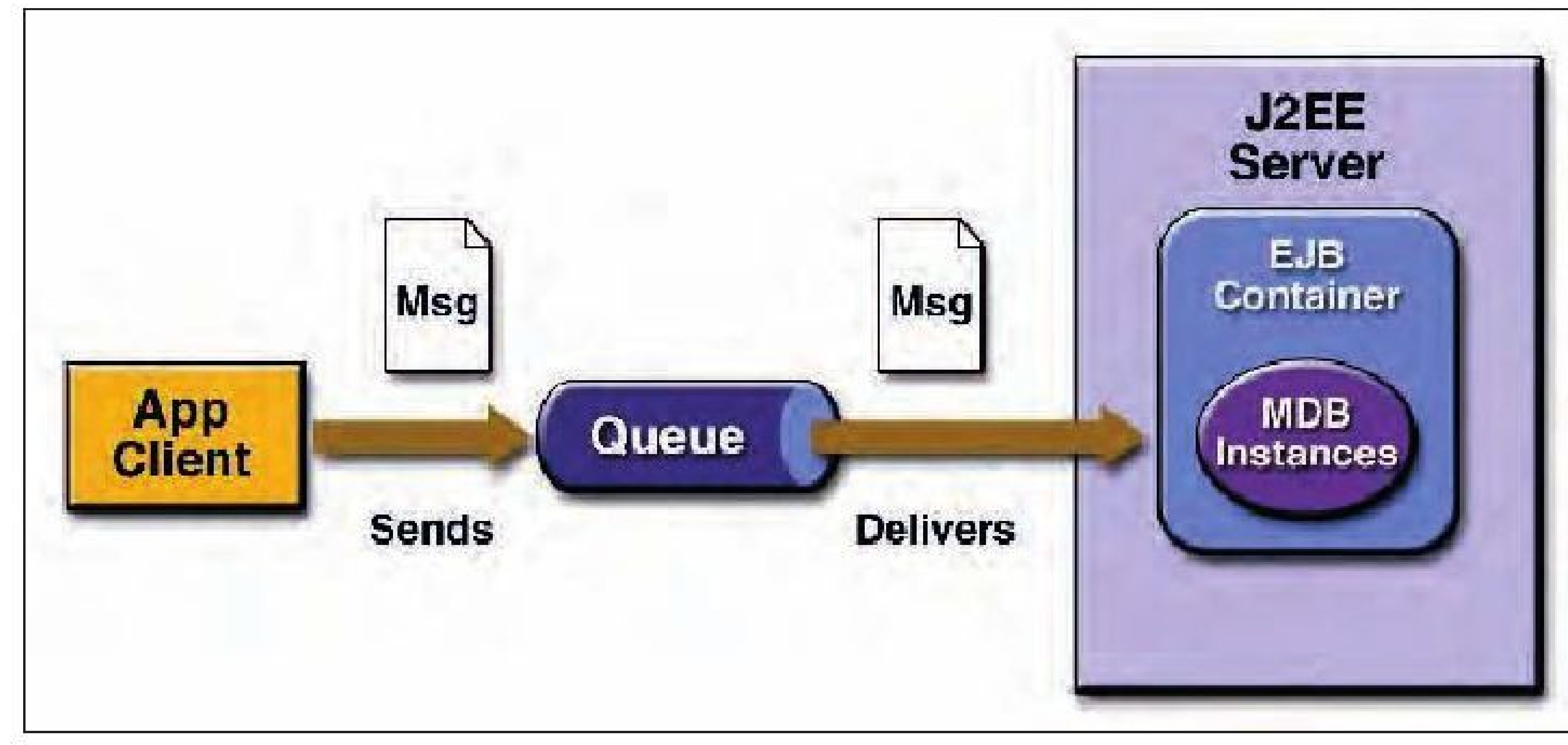


Figura 7. Colas EJB.



hecho de saber cuándo podremos tener disponible una reserva, puede que sea algo no trivial puesto que pueda depender de muchos factores; en este caso, simplemente basta con saber que la reserva ha sido hecha efectiva sin saber cuando podrá el cliente pasarse a recoger su pedido, estamos ante un proceso asíncrono. Estas llamadas asíncronas en Java son representadas mediante *JMS* y con *EJB*s orientados al mensaje con sus *colas* o *tópicos* asociados.
Llegados aquí hagamos un breve repaso; hemos descrito un sistema software a desarrollar y hemos definido una arquitectura de desarrollo del mismo basada en Java, a continuación vamos a describir de forma algo más concreta como vamos a trabajar en el desarrollo.

Herramientas de desarrollo

Como entorno de desarrollo vamos a utilizar *Eclipse*, sin entrar aquí en valoraciones ni comparaciones simplemente es el entorno que utilizo normalmente; en cualquier caso el código va a ser lo suficientemente general y no va a depender para nada del entorno de desarrollo; como tiene que ser claro está, de hecho el lector habrá notado que los diagramas *UML* están hechos con *Netbeans*.
Como requisitos previos, será necesario tener instalada la versión del *JDK de Java v1.6* y *Eclipse Europa*. Para la escritura y prueba de este artículo me encuentro trabajando en *Linux Kubuntu*, para Windows los requerimientos van a ser los mismos. Como base de datos vamos a utilizar *MYSQL* y como servidor de aplicaciones *JBOSS*.
Todo esto representa un "arsenal" tecnológico que, conviene recalcar, pone a disposición del programador los desarrolladores de estos productos Open Source, y que desde mi modesto punto de vista se merece un aplauso.
Para simplificar el ejemplo, las pruebas las realizaremos sobre un mismo ordenador físico, eso sí, va a quedar claro y explicado como realizar los test con varios ordenadores, esto nos servirá de paso, para explicar las diferencias entre "local" y "remoto".

JBOSS, Estructura y elementos de desarrollo

Una vez instalado *JBOSS* nuestra primera tarea va a consistir en configurar un servidor donde vayamos a probar nuestros proyectos; esto es algo bastante sencillo y además está explicado de forma amplia en la literatura sobre desarrollo; el lector conoce que bastará con duplicar uno de los directorios bajo: "$JBOSS_HOME/server/" por ejemplo y en nuestro caso, el "$JBOSS_HOME/server/default". Como el lector sabe, el servidor "default" conlleva de forma predeterminada una serie de servicios *J2EE*, que en nuestro caso van a ser suficientes para cumplir con todas las exigencias del proyecto. Para arrancar el servidor, bastará ir a la línea de comandos y ejecutar el comando "$JBOSS_HOME/bin/run", pasándole como parámetros el servidor que queramos arrancar, en nuestro caso será: "-c solop".
Quizá uno de los aspectos más importantes y que es necesario conocer, es como *JBOSS* puede conectarse a una base de datos; para hacerlo bien es a través de un DataSource. La conexión a una base de datos en *Java* es a través de interfaces *JDBC;* en un entorno de desarrollo profesional, como el que estamos describiendo aquí, es absolutamente necesario que nuestro código no contenga valores "hardcoded" o constantes que hagan que para cambiarlos sea necesario recompilar el mismo, es decir, deberán estos leerse desde algún fichero de configuración; este es una de las ventajas, por así decirlo de los DataSource, más adelante veremos como configurar uno.
Los DataSource son uno de los elementos que permiten configurar de forma declarativa y externa al código un servicio, pero no son los únicos, si observamos el directorio "$JBOSS_HOME/server/solop/conf", veremos unos cuantos más y que explicaremos más adelante como gestión de logs y gestión de usuarios para acceso a aplicaciones ("Realms") .
Como punto final a este mini resumen de *JBOSS* hay que conocer también el directorio "$JBOSS_HOME/server/solop/lib", que es donde hay que instalar o mejor dicho copiar las librerías externas al servidor que queremos que estén disponibles en el "classpath" de las aplicaciones que implantemos al arrancar el servidor, por ejemplo, aquí es donde irán los driver *JDBC* de *MYSQL*.

Eclipse, ANT, Estructura proyectos desarrollo

El entorno de desarrollo que vamos a utilizar a lo largo de este artículo es *Eclipse*, de todas formas no hay ningún problema para el lector que quiera utilizar *NetBeans*. Vamos a tener definido mediante un fichero independiente del IDE la forma de compilar, empaquetar, instalar, ... etc. nuestras aplicaciones. En Java existe una herramienta que nos viene como anillo al dedo y que es un standard de facto: *ANT;* esta herramienta, puede incluso ejecutarse sin la necesidad de un IDE, a través de la línea de comandos o desde una aplicación, por lo que podríamos desarrollar nuestro proyecto con nuestro editor de textos preferido; esto queda a la elección de cada desarrollador, aunque personalmente no es algo que yo aconseje.
Existen otras herramientas por así decirlo dentro de la categoría de *ANT*, por ejemplo, *MAVEN*, por favor entiéndame el lector a que me refiero; por su simplicidad en este artículo emplearemos *ANT* como digo, en cualquier caso creo que esta herramienta es de obligado conocimiento para un desarrollador *JAVA* profesional.

Conclusiones

Este es un artículo muy teórico donde hemos expuesto un "arsenal" de herramientas para la programación de un proyecto profesional en *JAVA*. Hemos hecho un resumen con todo lo que tiene que saber de antemano un arquitecto de desarrollo, puesto que de esta forma se podrá planificar el mismo.
Conviene tener en cuenta, puesto que es muy importante, ya no sólo aspectos concretos de tecnologías en si, sino como a través de modelos y mediante la aplicación de unos principios básicos de la ingeniería del software se desarrollan hoy en día los proyectos. Hemos cubierto todo lo que sería la planificación de un proyecto incluyendo la división de subproyectos para conseguir una mayor modularidad; hemos repasado una serie de tecnologías que nos permiten "desacoplar" aplicaciones en *JAVA* mediante llamadas asíncronas, hemos explicado la diferencia con respecto a las llamadas síncronas; hemos explicado las ventajas principales de Struts y como se integra junto a otra serie de tecnologías como la seguridad. Para finalizar, hemos definido mediante que servidor vamos a instalar y ejecutar los proyectos y un IDE con el que vamos a desarrollar los mismos.
En próximas entregas veremos las aplicaciones que hemos definido aquí desarrolladas tal y como hemos dicho y explicaremos más detenidamente aspectos sobre las tecnologías utilizadas.



Preguntas y respuestas

ADOLFO ALADRO GARCÍA

¿Cómo se manda un mensaje de correo electrónico con texto simple desde un *servlet* utilizando el *API* estándar de *Java*?

Antes de utilizar el *API* estándar contenido en el paquete *javax.mail* hay que verificar que los ficheros *mail.jar* y *activation.jar* se encuentran en el *classpath* del servidor de aplicaciones. Cada servidor de aplicaciones se configura de una forma distinta pero típicamente existe un directorio *lib* donde se guardan todas las librerías que utilice la aplicación *Web*. El siguiente ejemplo muestra cómo se puede mandar un mensaje de forma sencilla. El método *sendEmail* se define:

```
private final static void
sendEmail(String sSmtpServer, String
sTo, String sFrom, String sSubject,
String sTextContent)
```

El parámetro *sSmtpServer* es una cadena de texto que contiene el servidor *SMTP* al que la aplicación se conecta para enviar el mensaje. Los siguientes parámetros se corresponden con los datos típicos de un correo electrónico: la dirección de correo electrónico de los destinatarios (*sTo*), la dirección de correo electrónico del remitente (*sFrom*), el asunto del mensaje (*sSubject*) y el cuerpo del mensaje (*sTextContent*).

En el código del método estático *sendEmail* primeramente se obtiene una instancia de la clase *Properties* con las propiedades del sistema y se establece una nueva propiedad, denominada *mail.smtp.host*, con cadena de texto correspondiente al servidor *SMTP*:

```
Properties props =
System.getProperties();
props.put("mail.smtp.host",
sSmtpServer);
```

Obsérvese que el paso anterior puede evitarse si el servidor de aplicaciones ya arranca con las propiedades adecuadas configuradas. Además, una vez que se han establecido las propiedades del sistema no sería necesario volverlo a hacer.

La clase *Session*, del paquete estándar *javax.mail*, representa una sesión de correo electrónico. Existen diversas formas de obtener una instancia de esta clase. Una de ellas consiste en utilizar el método *getDefaultInstance*, el cual recibe como parámetro las propiedades en forma de objeto de tipo *Properties*:

```
Session session =
Session.getDefaultInstance(props);
```

Este método devuelve la sesión por defecto. Si la sesión todavía no ha sido establecida, se crea una nueva y se instala como sesión por defecto. Desde el punto de vista del rendimiento lo más habitual es que las aplicaciones compartan el mismo objeto *Session*.

En *Java*, el mensaje de correo electrónico se crea obteniendo una instancia de la clase *MimeMessage*:

```
Message message = new MimeMessage
(session);
```

El constructor recibe un único parámetro que se corresponde con la sesión de correo electrónico. La clase *MimeMessage* extiende a la clase abstracta *Message*, la cual sirve representa a todos los distintos tipos de mensajes de correo electrónico que pueden existir (en formato *HTML*, con ficheros adjuntos, etc.).

Antes de emplear los métodos de la clase *Message* para configurar el correo electrónico es preciso obtener el remitente (*sTo*) y los destinatarios (*sFrom*) en forma de objetos de tipo *InternetAddress*:

```
InternetAddress[]
arrInternetAddressTo =
InternetAddress.parse(sTo, false);
```

```
InternetAddress internetAddressFrom =
new InternetAddress(sFrom);
```

El método estático *parse* de la clase *InternetAddress* recibe dos parámetros. El primero es una cadena de texto que contiene la dirección de los destinatarios del mensaje. Puede haber un único destinatario o puede hacer varios. Cuando éste es el caso el segundo parámetro del método *parse* indica cómo se interpreta la cadena de texto. Un valor *false* establece que la cadena *sTo* no se interpreta de forma estricta siguiendo el estándar *RFC822* así que por ejemplo las direcciones de correo electrónico pueden estar separadas por blancos. El remitente del mensaje es único y para obtener el correspondiente objeto *InternetAddress* simplemente se utiliza el constructor de la clase.

El mensaje de correo electrónico finalmente se configura utilizando los métodos *setRecipients*, *setFrom*, *set Subject* y *setText*:

Overview Package Class Tree Deprecated Index Help
PREV CLASS NEXT CLASS — FRAMES NO FRAMES All Classes
SUMMARY: NESTED | FIELD | CONSTR | METHOD — DETAIL: FIELD | CONSTR | METHOD

javax.mail
Class Transport

java.lang.Object
└ javax.mail.Service
 └ javax.mail.Transport

public abstract class **Transport**
extends Service

An abstract class that models a message transport. Subclasses provide actual implementations.

Note that Transport extends the Service class, which provides many common methods for naming transports, connecting to tran
connection events.

Version:
1.36, 03/04/10
Author:
John Mani, Max Spivak, Bill Shannon
See Also:
Service, ConnectionEvent, TransportEvent

Documentación de la clase Transport del API estándar de Java para mandar mensajes de correo electrónico.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <!-- ************************** -->
    <!-- tagstr-tokenizer template -->
    <!-- ************************** -->
    <xsl:template name="tagstr-tokenizer">
        <xsl:param name="tagsstr"/>
        <xsl:variable name="separator">
            <xsl:choose>
                <xsl:when test="contains($tagsstr, ' ')">
                    <xsl:text>1</xsl:text>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:text>0</xsl:text>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:variable>
        <xsl:variable name="token">
            <xsl:choose>
                <xsl:when test="$separator = '1'">
                    <xsl:value-of select="substring-before($tagsstr, ' ')"/>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:value-of select="$tagsstr"/>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:variable>
        <xsl:variable name="remaining">
            <xsl:choose>
                <xsl:when test="$separator = '1'">
                    <xsl:value-of select="substring-after($tagsstr, ' ')"/>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:value-of select="string('')"/>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:variable>
```

Documentación de la clase Transport del API estándar de Java para mandar mensajes de correo electrónico.

```
message.setRecipients(Message.
RecipientType.TO, arrInternetAddressTo);
message.setFrom(internetAddressFrom);
message.setSubject(sSubject);
message.setText(sTextContent);
```

El último paso es el envío del mensaje propiamente dicho. Para ello se emplea la clase abstracta *Transport*, perteneciente al paquete *javax.mail*. El método estático *send* recibe un objeto de tipo *Message* y manda el correo electrónico:

```
Transport.send(message);
```

Desarrollo una aplicación *Web* que emplea *XSLT*. En los *XML* de origen hay un elemento que almacena de una cadena de texto con etiquetas (*tags*). Las etiquetas están separadas por blancos. En la página *HTML* resultante, con *XSLT*, tienen que salir las etiquetas enlazadas individualmente: ¿cómo se trocea la cadena original con *XSLT* para formar los enlaces?

En *XSLT* no hay bucles en el sentido clásico de la programación así que este tipo de problemas requieren una solución recursiva utilizando plantillas (*templates*) que se llaman a sí mismas. La plantilla cuya definición se muestra seguidamente se denomina *tagstr-tokenizer* y recibe un único parámetro denominado *tagsstr*, que típicamente será la cadena de texto que contiene las etiquetas separadas por blancos:

```
<xsl:template name="tagstr-tokenizer">
  <xsl:param name="tagsstr"/>
  ...
</xsl:template>
```

El primer paso consiste en determinar si en la cadena de texto hay una única etiqueta o varias.

Para ello se busca el espacio en blanco y finalmente la variable (en el sentido en el que la palabra variable se emplea en *XSLT*, que es bien distinto del concepto clásico de variable en programación) *separador* tiene 1 ó 0 dependiendo de si se encuentra o no:

```
<xsl:variable name="separator">
  <xsl:choose>
<xsl:when test="contains($tagsstr, ' ')">
      <xsl:text>1</xsl:text>
    </xsl:when>
    <xsl:otherwise>
      <xsl:text>0</xsl:text>
    </xsl:otherwise>
  </xsl:choose>
</xsl:variable>
```

Si hay espacio en blanco significa que existen varias etiquetas. En ese caso la variable *token* contiene la primera de ellas, que se obtiene con la función *substring-before* de *XPath*. Si hay una única etiqueta entonces *token* guarda dicha etiqueta directamente:

```
<xsl:variable name="token">
  <xsl:choose>
    <xsl:when test="$separator = '1'">
      <xsl:value-of select="substring-
before($tagsstr, ' ')"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$tagsstr"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:variable>
```

La variable *remaining* contiene la cadena restante una vez eliminada la primera etiqueta, si es que existían varias, o la cadena vacía si sólo había una:

```
<xsl:variable name="remaining">
  <xsl:choose>
    <xsl:when test="$separator = '1'">
      <xsl:value-of select="substring-
after($tagsstr, ' ')"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="string('')"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:variable>
```

La plantilla genera ahora el correspondiente enlace para la primera etiqueta empleando el valor almacenado en *token*:

```
<a href="http://misitio.com?tag={$token}">
  <xsl:value-of select="$token"/>
</a>
```

El último paso ilustra el verdadero truco de la solución. Si la variable *remaining* con lo que queda de la cadena tiene una longitud mayor que cero la plantilla se llama a sí misma con ella como parámetro:

```
<xsl:if test="string-length
($remaining) &gt; 0">
  <xsl:text disable-output-
escaping="yes"><![CDATA[ ]]></xsl:text>
   <xsl:call-template name="tagstr-
tokenizer">
     <xsl:with-param name="tagsstr"
  select="$remaining"/>
   </xsl:call-template>
</xsl:if>
```

El resultado es que se crean tantos enlaces individuales como etiquetas separadas por un blanco hay.



Tiro a la diana

Jorge Rubira

En esta ocasión, vamos a crear un juego de tiro a diana desarrollado en Java en el que van apareciendo y desapareciendo dianas por la pantalla, a las que hay que disparar utilizando el ratón.

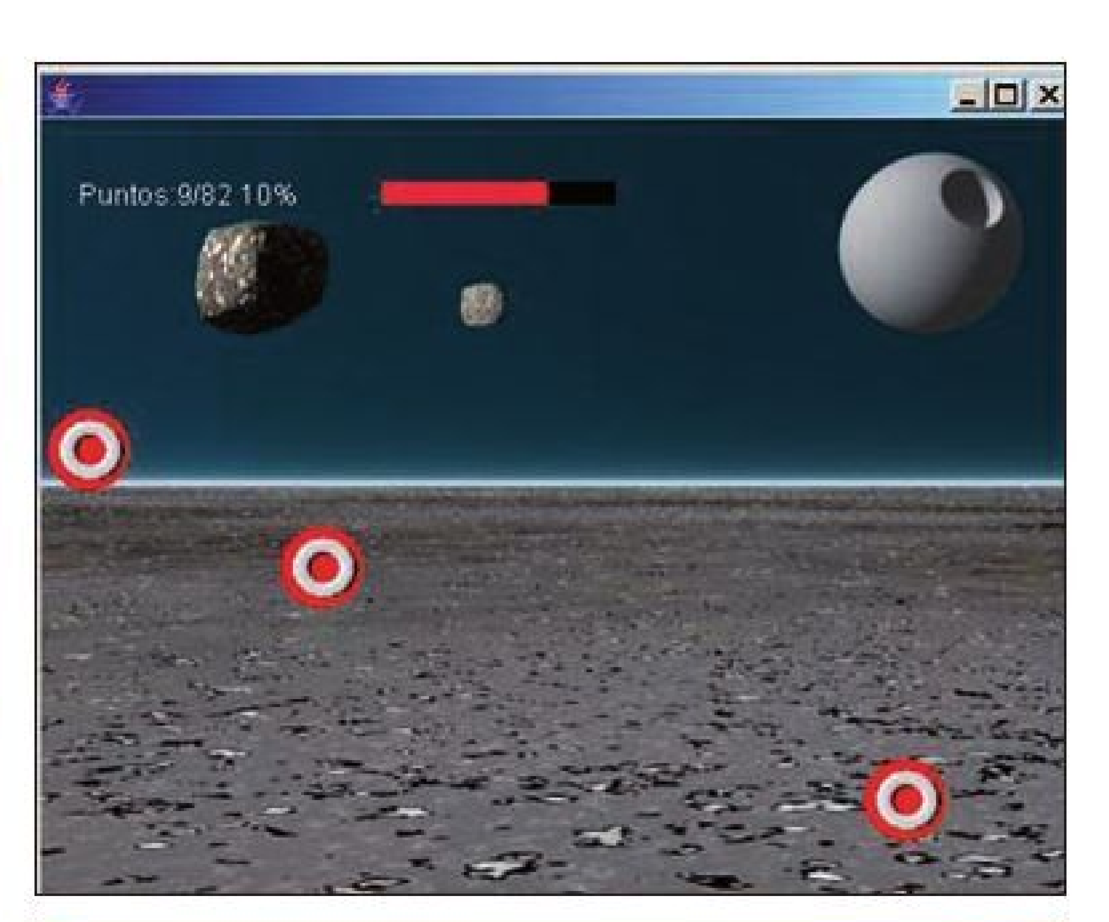
Las herramientas que utilizaremos en esta ocasión serán SDK 5.0 y Eclipse en su versión Europa. Jorge nos enseñará paso a paso cómo cargar, configurar y utilizar las distintas aplicaciones, y el paso a paso de creación y depuración del juego.

¿Cómo obtener el vídeo-tutorial?

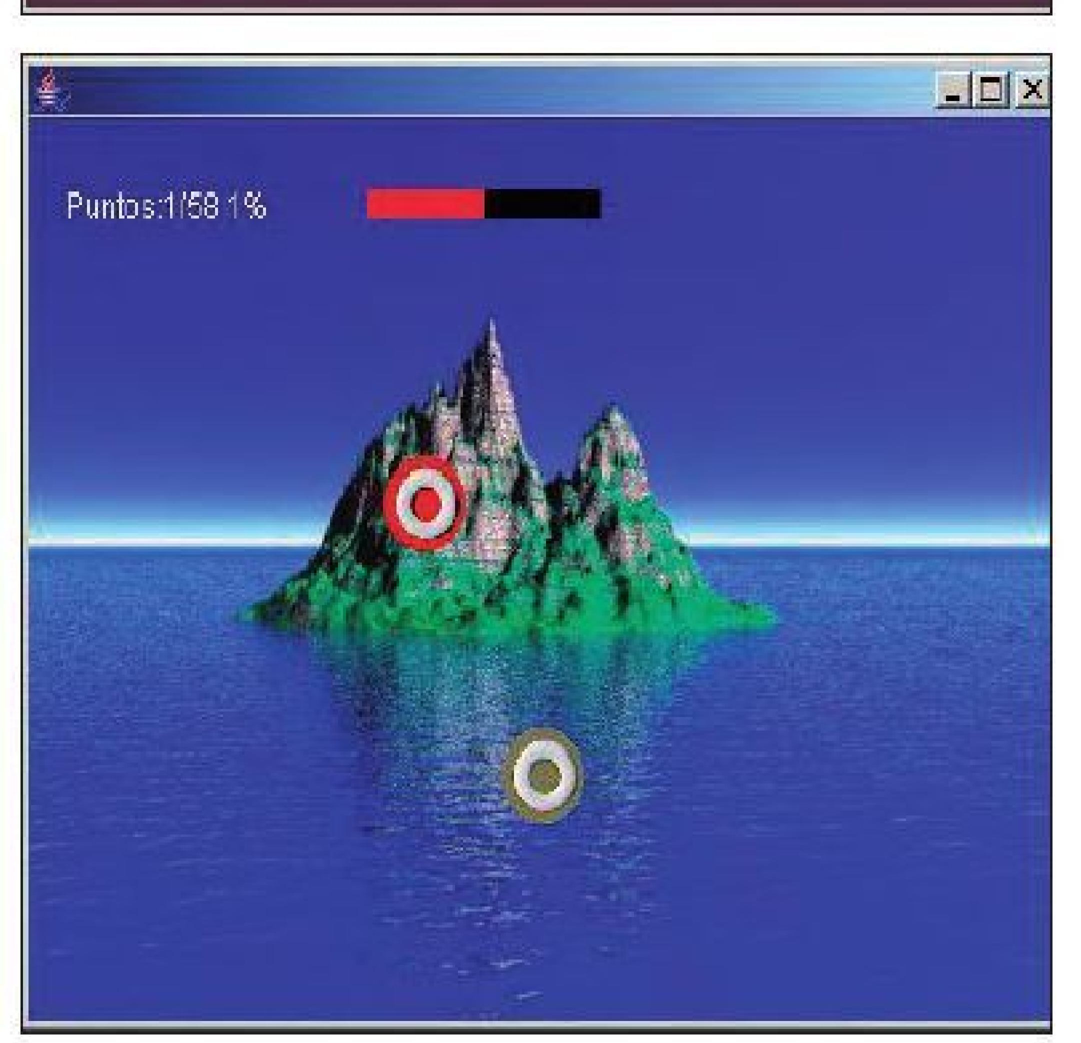
El material que conforma este vídeo-tutorial consiste en un archivo de vídeo y unos archivos de código que implementan el proyecto. Los lectores de la edición en papel encontrarán el vídeo-tutorial en el CD-ROM, mientras que los lectores de la edición digital lo encontrarán en el paquete descargado. Recordamos a los lectores de la edición digital que la nueva dirección es http://www.revistasprofesionales.com.
El formato del vídeo es WMV y el tamaño de la descarga es de 30 MB aproximadamente.











¿Tienes sugerencias para el próximo vídeo-tutorial?

Nos interesa saber cómo podemos mejorar los vídeos y sus contenidos, de modo que si tenéis cualquier sugerencia para futuros vídeos, no dudéis en transmitirla a Jorge Rubira, el autor: encuestavideos@gmail.com.

NÚMEROS ATRASADOS



160 - Mayo 2008

Veremos cómo acceder a los mecanismos de almacenamiento y haremos un repaso rápido a algunas librerías opcionales de Android. RSS se ha convertido en una tarea común en el desarrollado de aplicaciones Web, estudiaremos cómo hacerlo con Java. En este segundo artículo, seguiremos estudiando las capacidades de J2ME Polish. Exploramos las tecnologías que constituyen lo que se denomina como "web semántica con minúsculas". JavaCup 2008, segunda edición torneo de fútbol virtual Java. Hibernate y la sencillez de la capa de persistencia en JAVA. Con Java Media Framework (JMF) resulta muy fácil crear aplicaciones de vídeo y audio. 1 CD-ROM incluido. Incluye Solop 158 en PDF.



159 - Abril 2008

Paseo por The Evolution Show organizado por Microsoft para presentar sus últimos productos. Solución a problemas habituales desarrollando en J2ME pero que pueden mitigarse con J2ME Polish. Cómo generar un entorno de aprendizaje virtual a través del cual alumnos y profesores puedan interaccionar. Nuevas extensiones de Microsoft para ejecución paralela en .NET. Tratamos el Desarrollo de aplicaciones para redes sociales con OpenSocial de Google. Analizamos Java Media Framework (JMF) un software que permite crear aplicaciones Java. 1 CD incluído.



157 - Febrero 2008

Desvelamos las principales novedades de Visual Basic 2008. Análisis sobre la versión final del Service Pack 1 de Windows Vista. Análisis de Microsoft Visual Studio 2008, con la edición Express. Profundizaremos algo más en los tipos de aplicaciones que pueden hacerse con Android y las APIs disponibles. Descripción de PyS60 versión de Python ideada para dispositivos móviles con sistema operativo Symbian y de tipo Serie 60. Análisis de LINQ para SQL. Creación y gestión de componentes, así como la publicación de nuestras bases de datos en la web. Ponemos en práctica mediante la framework Jena las tecnologías semánticas revisadas en la primera parte. 1 DVD incluido.



157 - Febrero 2008

Nos sumergimos en el mundo Android, la nueva plataforma móvil de la Open Handset Alliance, para analizar las herramientas incluidas en su SDK y poder crear nuestra primera aplicación Android. Además, continuamos nuestro seguimiento a la tecnología 4D v11 SQL, iniciamos una serie dedicada a LINQ para SQL, continuamos nuestro desarrollo para iPhone, programamos la web semántica con Jena y ofrecemos la segunda entrega de nuestro curso AJAX. 1 CD-ROM incluido.



156 - Enero 2008

El iPhone ha irrumpido con fuerza en el mercado. En este número desvelamos las principales técnicas para programar aplicaciones web adaptadas a este innovador dispositivo. Además, entregamos la última parte del curso dedicado a la creación de buscadores con Lucene, una extensa revisión a la tecnología LINQ para XML, hacemos una primera introducción a Silverlight. Además, empezamos una serie avanzada sobre programación AJAX, en la cual crearemos un slideshow de imágenes. En términos de bases de datos, empezamos un tutorial sobre 4D v11 SQL, además de las secciones habituales. 1 DVD incluido.

Si te falta algún número de la temporada, ahora tienes la oportunidad de conseguirlo

Precio Oferta descuento

Precio por ejemplar: 6€

1 a 10 = 10% dto. / 11 a 20 = 20% dto.
21 a 30 = 30% dto. / 31 a 40 = 40% dto.
+40 = 50%

BOLETÍN DE PEDIDO

Rellene o fotocopie el cupón y envíelo a **REVISTAS PROFESIONALES, S.L.**
(Revista **SÓLO PROGRAMADORES**) C/ Valentín Beato, 42 - 3ª Planta - 28037 MADRID
Tel.: 91 304 87 64 - Fax: 91 327 13 03 - www.revistasprofesionales.com - rpsuscripciones@revistasprofesionales.com

Deseo me envíen los número/s: . . . .
NOMBRE Y APELLIDOS . . . . .EDAD . . . . TELÉFONO . . . .
DOMICILIO . . . . .C.P.: . . . .
CIUDAD . . . . .PROVINCIA . . . .

FORMAS DE PAGO

- ❑ Giro Postal a nombre de REVISTAS PROFESIONALES, S.L.
- ❑ Talon Bancario a nombre de REVISTAS PROFESIONALES S.L.
- ❑ Domiciliación Bancaria
  - Banco . . . .
  - Domicilio . . . .
  - Numero de cuenta: _ _ _ _/_ _ _ _/_ _/_ _ _ _ _ _ _ _ _ _ _ _
  - Titular . . . .
- ❑ Contra Reembolso (5€ de gastos de envio por paquete)
- ❑ Tarjeta de crédito _ _ _ _/_ _ _ _/_ _ _ _/_ _ _ _/ Fecha de caducidad:

Firma:

Extranjero: Gastos de envio 5€ por paquete. Unica forma de pago tarjeta de crédito (VISA, Mastercard, American Express,...)



Fuentes

ASP.NET (MVC)

Las primeras versiones fueron básicas, pero ahora ya se cuenta con una infraestructura bastante completa de modelo y controlador y vista (MVC)

RSS con Java II

Las aplicaciones Web de los servicios y portales que ofrecen RSS pueden verse sometidas a un gran estrés debido a la gran cantidad de documentos que tienen que servir simultáneamente. Por ello es indispensable implementar políticas de optimización que hagan que las aplicaciones puedan soportar esta carga con unos niveles aceptables de rendimiento.
Fundamentalmente se pueden optimizar dos tareas: la de creación de los documentos RSS y la de servirlos. Para lo primero se va a estudiar de qué manera puede mejorarse la clase RssDocument con el fin de evitar la creación constante de objetos que son susceptibles de poder ser reutilizados. Para lo segundo se va a analizar la creación de sistemas de caché que eviten tener que crear el mismo documento RSS repetidas veces en un corto espacio de tiempo.

Podcast javaHispano

Edición número 11 del podcast de Javahispano

Este podcast no está dedicado a una tecnología, si no a una persona bastante veterana y con mucha experiencia en el mundo del desarrollo de software. Hablamos de Francisco Morero Peyrona quien debemos agradecer que se ofreciese a responder preguntas de carácter general aunque al mismo tiempo muy subjetivas y complicadas de responder. Por otra parte tendremos la habitual sección de noticias donde Abraham y Alfredo nos hablarán de las tendencias en el mundo java.
Sección noticias: Presentado por Abraham Otero y Alfredo Casado:

- Reproductor de videos de Youtube basado en J2ME.
- Intellij IDEA, el primer entornoJava en reconocer que no está solo en el mundo.
- SpringSource Application Management Suite beta.
- Es Groovy lento? importa?
- JBuilder 2008.
- Microsoft planea mejorar Eclipse.
- JNIEasy 1.2.1 Soporte de Solaris x86 y Mac Leopard.

En la segunda parte del podcast entrevistaremos a Francisco Morero Peyrona donde nos dará su opinión, por supuesto subjetiva, sobre el desarrollo de software.



Edición número 12 del podcast de Javahispano

En este número 12 de Javahispano Podcast, hemos preparado un especial en el que hemos participado simultaneamente dos podcast: Linuxhispano Podcast y Javahispano Podcast. Además tendremos las noticias más actuales del portal.
Sección noticias: Presentado por Abraham y Alfredo.

- JavaCup 2008
- El servidor web Jetty en claro crecimiento
- Nueva era de hielo!! - ICEfaces 1.7.0
- Java ME CDC ejecutándose en el iPhone
- Conozcamos PulpCore!
- IBM anuncia Project Zero, un stack al estilo Ruby on Rails
- Glassfish v3 adopta OSGi

Sección tertulia: Especial realizado conjuntamente Javahispano y Linuxhipano. Hablaremos junto nuestros compañeros de LinuxHispano Podcast de Java, Linux y Licencias Libres. Presentado por Nacho Lopez, Javier Carazo, Erick Camacho y Jorge Rubira.

Vídeo – Tutorial

En esta ocasión, vamos a crear un juego de tiro a diana desarrollado en Java en el que van apareciendo dianas a las que hay que disparar utilizando el ratón.
Las herramientas que utilizaremos en esta ocasión serán SDK 5.0 y Eclipse en su versión Europa.
Jorge nos enseñará paso a paso cómo cargar, configurar y utilizar las distintas aplicaciones, y el paso a paso de creación y depuración del juego.

Además ...

Vídeos e información sobre JavaCup.
Solo Programadores 159 en formato pdf.

